DATABASE MANAGEMENT SYSTEM

Authors	Gaurav Kansal, B P Sharma, Pratibha Singh, Nidhi Singh, Anand Kr. Srivastava, Ritin Behl, Sachin Goel
Authorized by	
Creation/Revision Date	March 2021
Version	1.0

UNIT -1
INTRODUCTION

Table of Contents

1. Introduction

Every organization works around five resources, generally referred to as 5Ms named Man, Machine, Material, Money, and Methods.

Here Man refers to the human resource or the employees in the organization, machine refers to the machinery used to perform various tasks in the organization, material refers to the raw material and finished goods produced in the organization, money refers to the financials available in the organization in various forms and Methods refers to the way an organization works.

If the organizations do not have accurate information about these five resources available to them, they cannot work efficiently; that is why information is the 6^{th} resource.

The information cannot be generated about these resources until we have the relevant data about these resources like how many people we have in our organization, what is their expertise, how much money we have with us to invest, what kind of machines we have, and how to accomplish a particular task.

Data is the raw material to produce information, which is the most valuable input for organizations today to do the effective decision making. It is generally kept under high security, just like cash.

Data can be managed in various formats depending on the type of data, type of security required (high, low, medium), type of access required (like web, mobile, desktop), and some more factors.

This chapter aims to introduce the data, information, and data management techniques using specialized systems called Relational Database Management System (RDBMS).

1.1 Data and Information

1.1.1 Data

Data are the facts that can be recorded in the form of texts, numbers, videos, audio, or images.
For example:-

- Roll number, student name, course name, branch name, date of birth, admission date are the facts about a student.
- Book number, book title, author name, publisher name, purchase date, and price are the facts about a book in a library.

- Location, date, time are the facts managed by Google Maps while we move from one place to another.
- Videos on YouTube are the facts posted/hosted by a diverse set of users and for a diverse set of viewers.

Each piece of fact is technically called a ***datum***, and the plural of datum is called ***data***.
The collection of related data fields for a particular individual or item is called a **data record**.

For example:-

Roll number	Student Name	course name	branch name	date of birth	admission date
101	Ram	B.Tech	CSE	12-May-2001	03-Jun-2020
102	Shyam	MCA	NA	16-Apr-2002	04-Apr-2021

Figure 1. 1: Representation of Data Record

In the Figure 1.1, Roll number, student name, course name, branch name, date of birth, admission date about a student say Ram is an example of Data Record.

1.1.2 Information

Data in itself has very limited utility when it comes to decision-making. Data needs to be processed using a set of procedures to convert it into information. This information can now be used for the desired purpose. The information will always be need-based or purpose-specific.

For Example:-
Roll number, student name, course name, branch name, date of birth, admission date are the facts about a student – which is "data".

Now, if we process this data to find – Number of students in a particular branch? Number of students below the age of 18 years? This is “Information”.

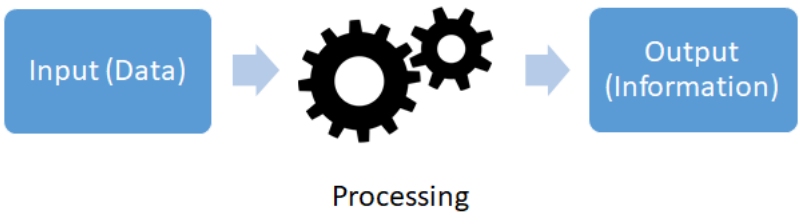


Figure 1. 2: Process of conversion of Data into Information

The information is user-need specific. Something which is information for one user can be just a piece of data for another and vice-versa.

If a student scores greater than 90 marks in specific course that can be indicator of the area of interest of that student. This represents the “Knowledge”.

An intelligent system can be build using the acquired knowledge to recommend the job to the student according to their area of interest. This represents an “Intelligence”.

Therefore, data plays a very important role in building intelligent systems. We cannot build intelligence systems until we have a correct and comprehensive set of data.

The relationship can be depicted in Figure 1.3 as:-

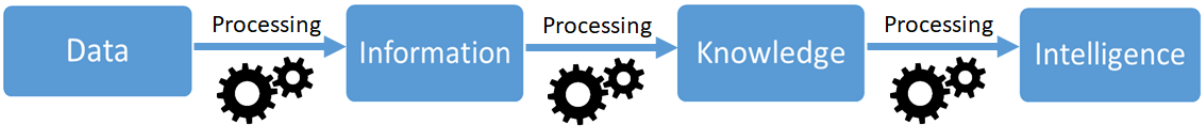


Figure 1. 3: Conversion of Data into intelligence

1.2 Database and Database Management System

1.2.1 Managing Data

We often need to access and re-sort data for various uses. These may include:

- Creating mailing lists
- Writing management reports
- Generating lists of selected news stories
- Identifying various client needs

To manipulate the data, we need to store the data in well-defined structures. These requirements, such as persistence of data, operations on data, etc., prompted the development of database systems.

1.2.2 Database

A **database** is an organized collection of related data, typically stored in a computer system. For example, a database that contains data about students should not hold data about company stock prices.

Here *organized* means, all data is described and associated with other data. All data in a database are *related* to each other; separate databases should be created to manage unrelated data.

A database is integral to any company or organization. This is because the database stores all the pertinent details about the company, such as employee records, business transactional records, salary details, inventory, etc.

1.2.3 Database Management System (DBMS)

Database Management System (DBMS) is the software that is used to manage the database. DBMS allows the users to carry out the following tasks:-

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

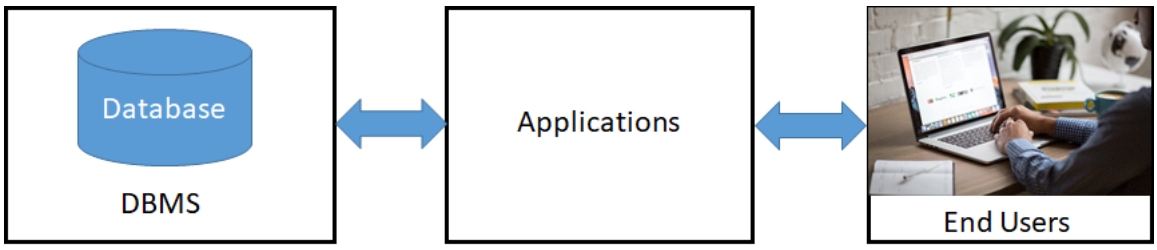


Figure 1. 4: Relationship between End User, Application, and DBMS

A database needs to be hosted or created on DBMS. Some of the famous DBMS are:

- PostgreSQL
- MySQL
- Microsoft Access
- SQLite

To access any data from a database through an application, we need the layer of DBMS. An application uses the DBMS functionalities to store the data and retrieve the data from database. The following images show how an application access the data from the database.

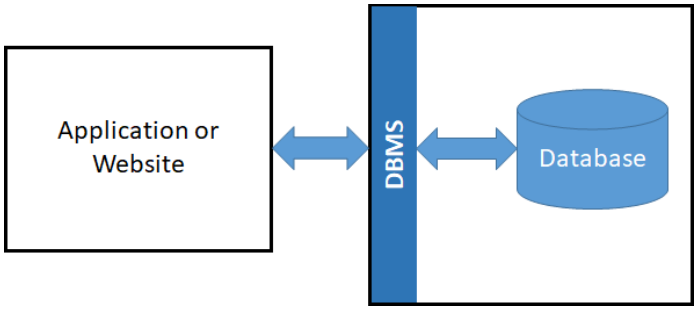


Figure 1. 5 : Application access data from the database via DBMS

1.2.4 File System Vs Database Management System

Why DBMS? Why not store data on file systems?

Let us consider the example of a file-based University Management System.
File systems consist of different files which are grouped into directories. The directories further contain other folders and files. The file system performs basic operations like management, file naming, giving access rules, etc.
E.g., NTFS (New Technology File System), EXT (Extended File System).
Students' data is available to their respective Departments, Academics Section, Result Section, Accounts Section, Hostel Office, etc. Some of the data is common for all sections like RollNo, Name, Father Name, Address, and Phone number of students, but some data is specific to a particular section only like Hostel allotment number part of hostel office. Let us discuss the issues with this system:

- **Redundancy of data:** Data is redundant if the same data is stored at multiple places. The student's common data in our case will be stored at multiple places across all the sections. This leads to increased use of memory and hence higher cost.
- **Inconsistency of Data:** Data is inconsistent if multiple copies of the same data do not match with each other. If a student wants to change his Phone number, he has to get it updated in various sections. If this doesn't happen, his Phone number might be different in Accounts Section and Academics Section. This will lead to data inconsistency. Inconsistency can occur because of not updating all copies of the same data or typing errors.
- **Difficult Data Access:** A user should know the exact location and format of the file to access and use data. It is a big challenge as the data might be needed by multiple users at multiple locations. File access management becomes very cumbersome and tedious.
- **Unauthorized Access:** File Systems do not have robust access control. This may lead to unauthorized access to data. If a student gets access to a file having his marks, he can change it in an unauthorized way.
- **No Concurrent Access:** The access of the same data by multiple users at the same time is known as concurrency. The file system does not allow concurrency as data can be accessed by only one user at a time.
- **No Backup and Recovery:** The file system does not incorporate any backup and recovery of data if a file is lost or corrupted.

A brief history of DBMS

The first general-purpose DBMS was designed by Charles Bachman at General Electric in the early 1960s, called the Integrated Data Store. It formed the basis for the network data

model, which was standardized by the Conference on Data Systems Languages (CODASYL) and strongly influenced database systems through the 1960s.

The other important events in history are mentioned below:

- 1960s: navigational databases, mostly hierarchical databases and network databases were first used
- 1970: E.F. Codd from IBM introduces the concept of relational databases and the first normal form
- 1971: second and third normal form are introduced by Codd
- 1973: project INGRES (the predecessor of PostgreSQL) begins in Berkeley
- 1974: project System R begins at IBM, developing the first implementation of SQL
- 1974: Boyce-Codd normal form is introduced
- 1976: the entity-relationship model is introduced by Peter Chen from MIT
- 1978: the first release of Oracle DB, based on IBM's papers
- 1979: public release of Oracle Version 2 (version 1 was never released)
- 1983: the very first release of IBM DB2 database
- 1985: project INGRES ends, the post-Ingres project starts in Berkeley
- 1990: the first public release of Postgres database
- 1990s: relational databases are the industry standard
- 2000s: NoSQL and NewSQL movements emerge

1.3 Database Design

1.3.1 Data Models

Database models:

The model represents a perception of reality. For example, a city map represents the model of the city in reality.

A data model is a collection of concepts that can be used to represent the structure of a database in a simple way. It provides a way to describe the design of the database at the physical, logical and view level.

A collection of conceptual tools for describing a data model are:

- Data
- Data relationships – Data Structure
- Data Semantics
- Data constraints

Importance of Data Models:

- Relatively simple representation, usually graphical, of complex real-world data structures
- Communication tools to facilitate interaction among the designer, the applications programmer, and the end-user
- Good database design uses an appropriate data model as its foundation
- End-users have different views and needs for data. Data model organizes data for various users
- It hides the complex details of its implementation.
- It helps to understand the complexities of the real-world environment.
- It improves understanding of the organization

The most prevalent database model for application development today is the Relational model. This will be our focus area for this course. We will also discuss two legacy data models – Hierarchical Model and Network model, briefly for completeness of discussion:

1. Hierarchical Model

Hierarchical Database Model, as the name suggests, is a database model in which the data is arranged in a hierarchical tree edifice. As it is arranged based on the hierarchy, every record of the data tree should have at least one parent, and each parent should have one or more child records, except for the child records in the last level. The Data can be accessed by following through the classified structure, always initiated from the Root or the first parent. Hence this model is named as Hierarchical Database Model.

Hierarchical database model is one of the oldest database models, which originated in 1950. One of the first hierarchical database model was Information Management System developed by North American Rockwell company and IBM.

In this model, data is stored in the form of records which are the collection of fields. The records are connected through links. Each field in a record can contain only one value. The first node of the tree is called the root node. When data needs to be retrieved, then the whole tree is traversed starting from the root node. This model can represent only one-to-many relationships. It is not able to represent many-to-many relationships. (Note:- we will study relationships in detail later in the course)

Let us take an example of college students who take different courses. A course can be assigned to a student, but a student can take as many courses as they want.

Example:

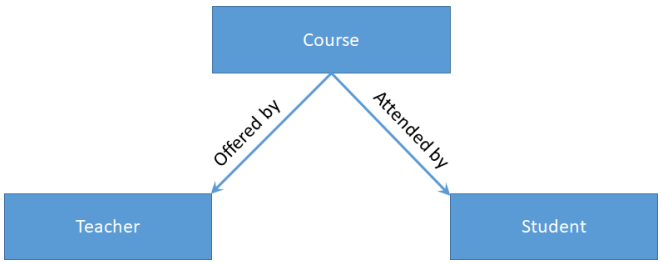


Figure 1. 6: Hierarchical structure

Records: Course, Teacher, Student

Relationship: Offered by, Attended by

Hierarchical model arranged as in Figure 1.6 indicates the relationship and can answer the queries like:

- Who is teaching a particular course?
- Who has registered in a particular course?

It does not indicate the relationships and cannot answer the following queries:

- What are the courses being offered by a faculty?
- What are the courses being attended by a student?
- Whom are the students being taught by a faculty?
- Whom are the faculty teaching a student?

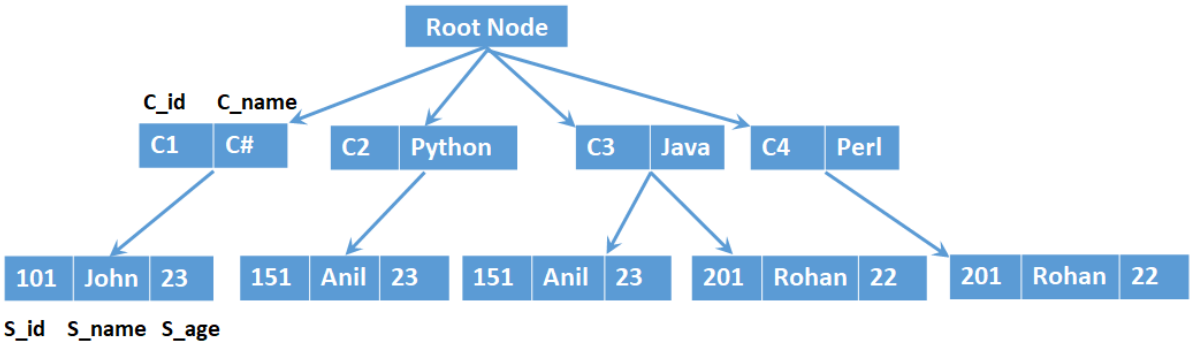


Figure 1. 7 (a): Example of Hierarchical structure (Course to Student)

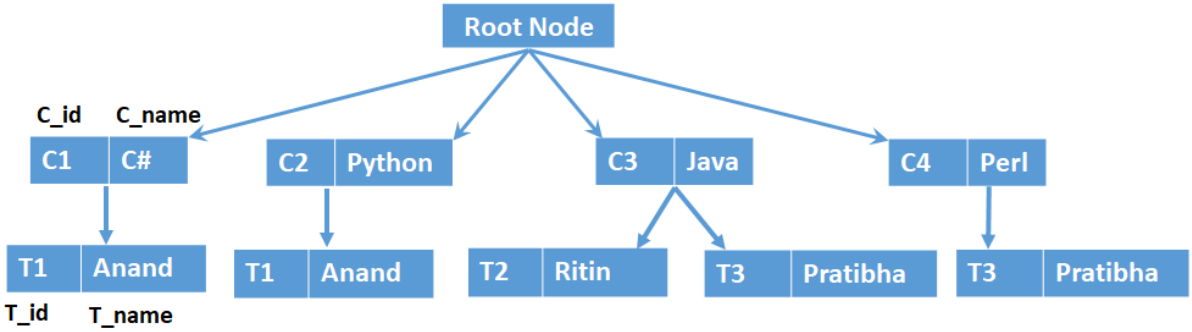


Figure 1.7(b): Example of Hierarchical structure (Course to Teacher)

As shown in Figure. 1.7(a), when one course is taken by more than one student, then we need to replicate the student record. As shown in the above example, Anil has registered for

courses Java and Python and thus, the record of Anil is repeated twice. Also, we can see that record of Rohan has also appeared twice in this model. It leads to the redundancy of data. The same hierarchical structure can be represented for the link course to teacher shown in Figure 1.7(b).

A Hierarchical database model was widely used during the Mainframe Computers Era. Today, it is used mainly for storing file systems and geographic information. It is used in applications where high performance is required, such as telecommunications and banking. A hierarchical database is also used for Windows Registry in the Microsoft Windows operating system. It is useful where the following two conditions are met:

1. The data should be in a hierarchical pattern, i.e., a parent-child relationship must be present.
2. The data in a hierarchical pattern must be accessed through a single path only.

Advantages:

- **Simplicity**: It is conceptually simple due to the parent-child relationship. A clear chain of command or authority is there. Data can be retrieved easily due to the explicit links present between the table structures.
- **Data Integrity**: Referential integrity is always maintained, i.e., any changes made in the parent table are automatically updated in a child table.
- **Data Security**: It is provided and enforced by DBMS.
- **Efficiency**: It is efficient when a large number of one-to-many relationships exist in a database

Limitations:

- A child record cannot be created if it is not linked to any parent record. In our example above, we cannot create a course until it is opted by any student.
- M: N relationship (many-to-many) is not supported.
- Redundancy can result in data inconsistency.
- Change in structure leads to change in all application programs.
- No data manipulation language or data definition language.
- Poor flexibility
- Rigid structure

2. Network Model

The network model replaces the hierarchical tree with a graph, thus allowing more general connections among the nodes. It allows a record to have more than one parent and multiple child records, thus permits many-to-many relationships. The network model was adopted by the Conference on Data System Language (CODASYL) Data Base Task Group in 1969 and underwent a major update in 1971. The operations on the network model are done with the

help of the circular linked list. A child node in the network model can have more than one parent record, as shown in Figure 1.8.

Example:

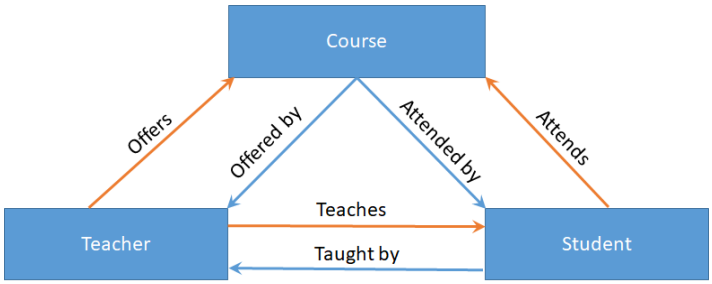


Figure 1. 8 : Network model

Records: Course, Teacher, Student
Relationship: Offered by, Attended by, Taught by, Offers, Attends, Teaches
Now the above network model can give the answer to all queries mentioned below, which was not possible in the case of Hierarchical model.

- What are the courses being offered by a faculty?
- What are the courses being attended by a student?
- Whom are the students being taught by a faculty?
- Whom are the faculty teaching a student?

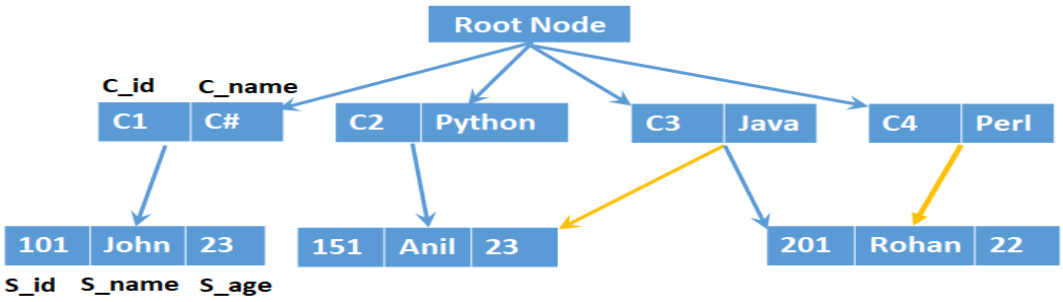


Figure 1. 9 : Example of Network structure (Course to Student)

An example of a network model, as shown in Figure 1.9, shows that the record of Anil has two parent records Python and Java. Similarly, the record of Rohan has two parent records of record type Course. It reduces the redundancy of data as depicted in Figure 1.7(a), showing hierarchical structure storing the same set of records.

Some of the popular network databases are:

- Integrated Data Store (IDS)
- IDMS (Integrated Database Management System)
- Raima Database Manager
- TurboIMAGE
- Univac DMS-1100

Advantages:

- **Conceptual Simplicity:** It is conceptually simple and easy to design.

- **Capability to handle more relationship types:** It handles many to many relationships, which is a real help in modeling real-life situations.
- **Ease of data access:** The data access is easier and flexible as compared to the hierarchical model.
- **Data Integrity**: It ensures data integrity as a user must define the owner record before defining the member record.

Limitations:

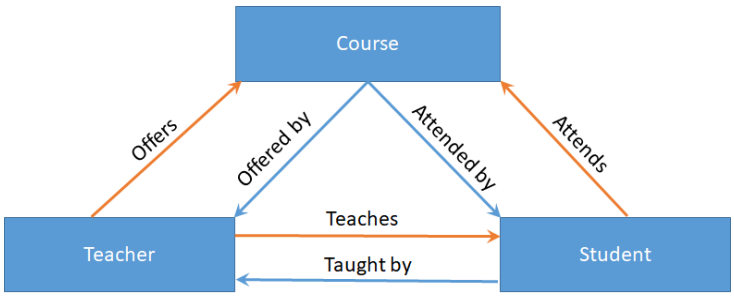
- **Implementation Complexity:** It requires the knowledge of actual physical data storage for navigational data access. The database administrators, designers, programmers, and even the end-users should be familiar with the internal data structure. It can't be used to create user-friendly DBMS.
- **Absence of structural Independence:** The data access method is navigational, which makes structural change very difficult and impossible in most cases. So if changes are made in the database structure, then all the application programs need to be modified.
- **Any change like updation, deletion, insertion is very complex.**

The data operation like retrieval, deletion, and insertion is very complex because of the graph structure. The network model uses the graph structure for data storage.

3. Relational Model

The relational model was introduced in 1970 by E. F. Codd (of IBM) in his landmark paper "A Relational Model of Data for Large Shared Databanks" (Communications of the ACM, June 1970, pp. 377−387). The relational model foundation is a mathematical concept known as a relation. A relation is also called a table, as a matrix composed of intersecting rows and columns. These tables represent both data and relationships amongst the data. Each table corresponds to an application entity. Each row in a relation is called a tuple and represents an instance of that entity. Each column represents an attribute. The relational model also describes a precise set of data manipulation constructs based on advanced mathematical concepts.

The relational data model is implemented through a very sophisticated relational database management system (RDBMS). The RDBMS performs the same basic functions provided by the hierarchical and network DBMS systems, in addition to other functions. Figure 1.10, shows a relational model corresponding to the network model of records Course, Teacher and Student.



Relation	Attributes
Student :	(S_id, S_name, S_age)
Course :	(C_id, C_name)
Teacher :	(T_id, T_name)
Teacher Course :	(T_id, S_id)
Student Course:	(S_id, C_id)

Student Relation

S_id	S_name	S_age
101	John	23
151	Anil	23
201	Rohan	22

Course Relation

C_id	C_name
C1	C#
C2	Python
C3	Java
C4	Pearl

Teacher Relation

T_id	T_name
T1	Anand
T2	Ritin
T3	Pratibha

Student Course Relation

S_id	C_id
101	C1
151	C2
151	C3
201	C3
201	C4

Teacher Course Relation

T_id	C_id
T1	C1
T1	C2
T2	C3
T3	C3
T3	C4

Figure 1. 10 : Example of Relational Model

Some popular Relational Database management systems are:

According to DB-Engines, in January 2021, the most widely used systems were Oracle, *MySQL (free software), Microsoft SQL Server, PostgreSQL* (Open Source, a continuation development after INGRES), IBM DB2, SQLite (free software), *Microsoft Access, MariaDB* (free software), *Teradata, Microsoft Azure SQL Database, Apache Hive* (free software; specialized for data warehouses)

According to research company Gartner, in 2011, the five leading proprietary software relational database vendors by revenue were *Oracle (48.8%), IBM (20.2%), Microsoft (17.0%), SAP including Sybase (4.6%), and Teradata (3.7%).[24]*

Advantages:

- Structural independence is promoted using independent tables. Database users need not be familiar with internal data storage.
- The tabular view improves conceptual simplicity. Database designers need to focus on logical design rather than knowing the physical data storage details.
- Ad hoc query capability is based on SQL. SQL is 4GL which permits the user to specify only what needs to be done without specifying how it will be done.
- Isolates the end-user from physical-level details
- Improves implementation and management simplicity
- Precision: The usage of relational algebra and relational calculus in the manipulation.

Limitations:

- Requires substantial hardware and system software overhead: It requires more powerful hardware and also physical storage devices to hide the complexity of implementation and data storage.
- Conceptual simplicity gives untrained people the tools to use a good system poorly. Poor database design degrades the performance when the database grows and it slows down a system.
- May promote information problems, i.e., information island phenomenon. As relational data model is easy to use, thus many departments may create their own database application, which creates the problem in integrating information and may create the issues of data inconsistency, redundancy, duplication, etc.
- Few relational databases have limits on the length of fields. It cannot be exceeded to accommodate more information.

There are some database model that extends the concept of object-oriented programming are discussed below in brief. The detailed discussion on these models is beyond the scope of this book.

Object-Oriented Data Model

Relational database models may fail to handle the needs of complex information systems. The problem with RDBMS is that they force an information system to model into form of tables. Increasingly complex real-world problems demonstrated a need for a data model that more closely represented the real world. In the object-oriented data model (OODM), both data and their relationships are contained in a single structure known as an object. Object oriented models represents an entity as a class and its instance as an object. A class represents both attributes and the behavior of an entity. Object-oriented data models can typically be represented using Unified Modeling Language (UML) class diagrams. It is used to represent data and their relationships. Some object-oriented databases are designed to work well with object-oriented programming languages such as Delphi, Ruby, Python, JavaScript, Perl, Java,

C#, Visual Basic .NET, C++, Objective-C and Smalltalk; others such as JADE have their own programming languages

Advantages:

a. Capability to handle large number of different data types. It can store any type of data including text, numbers, pictures, voice and video.
b. Object oriented programming with database technology
c. Semantic content is added.
d. Object oriented features improve productivity: Inheritance, Polymorphism and Dynamic binding

Limitations:

a. Difficult to maintain
b. Not suited for all applications.

Object-Relational Data Model

Object relational systems combine the advantages of modern object oriented programming languages with relational database features such as multiple views of data and a high-level, non-procedural query language. Relational database systems are being enhanced by adding an object infrastructure to the database system itself, in the form of user defined data types, functions and rules and by building relational extenders that support specialized applications as image retrieval, advanced text searching and geographic applications. PostgreSQL is an Object-Relational Database Management System.

Advantages:

a. Complex data sets can be saved and retrieved quickly and easily.
b. Works well with object-oriented programming languages.

Limitations:

a. Object databases are not widely adopted.
b. In some situations, the high complexity can cause performance problems.
c. High system overhead slows transactions

1.3.2 Database System Architecture

The database system architecture is divided into three levels:

- External or View level: Describes various user views
- Conceptual or Logical Level: Structure and constraints for the entire database
- Internal or Physical Level: Actual Physical storage structure and access paths

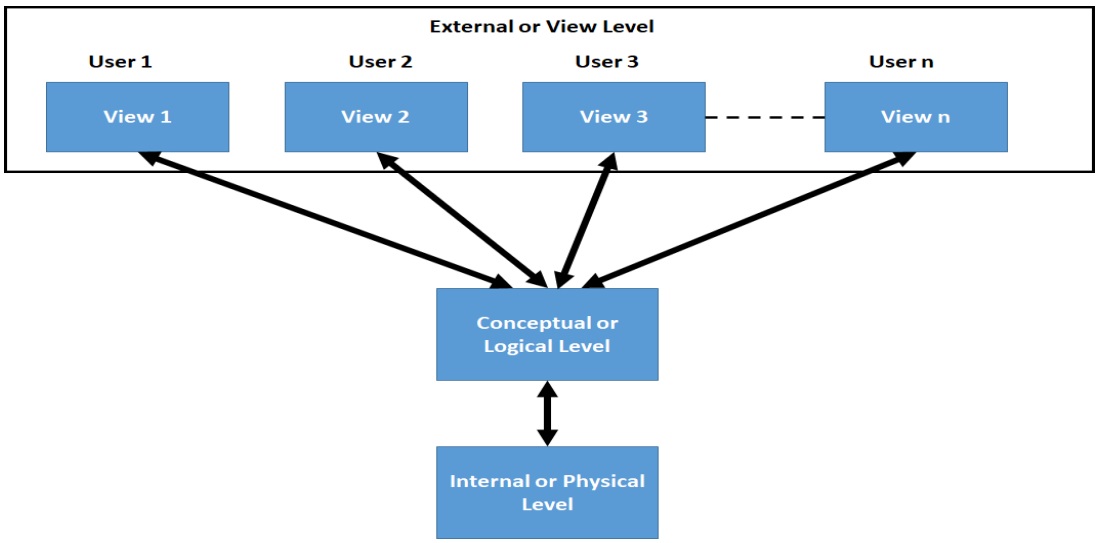


Figure 1. 11 : Database System Architecture

External level

It is also called **view level**. The reason this level is called "view" is that several users can view their desired data from this level which is internally fetched from the database with the help of conceptual and internal level mapping.

The user doesn't need to know the database schema details such as database structure, table definition, etc. The user is only concerned about data that is returned to the view level after it has been fetched from the database (present at the internal level).

Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking at the course details of students; STUDENTS are interested in looking at all details related to academics, accounts, courses, and hostel details as well. So, different views can be generated for different users. The main focus of the external level is data abstraction.

The external level is the "**top-level**" of the Three Level DBMS Architecture.

Conceptual level

It is also called the **logical level**. The conceptual schema describes the Database structure of the whole database for the community of users. This schema hides information about the physical storage structures and focuses on describing entities (tables) and their relationships, attributes (data elements) and their data types & constraints, user authorization, and authentication, etc.

For example, the STUDENT database may contain STUDENT and COURSE tables that will be visible to users, but users are unaware of their storage.

Internal level

This level is also known as **the physical level**. This level describes how the data is actually stored in the storage devices like discs and tapes. This level is also responsible for allocating space to the data. This is the lowest level of architecture.

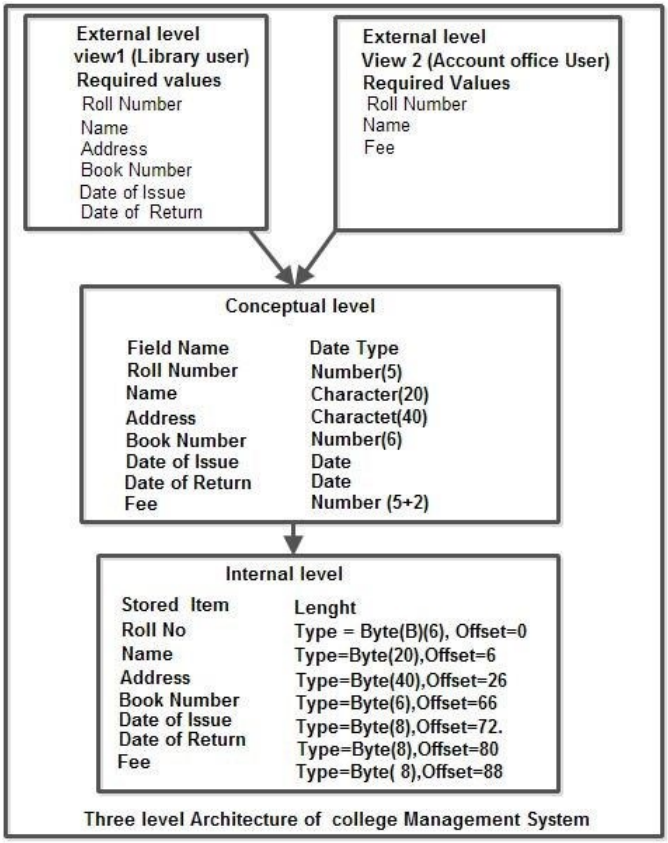


Figure 1. 12 : Three Level Architecture of Management System

Data Independence

The main objective of the 3-level architecture is to provide Data Independence, which means that upper levels are unaffected by changes at lower levels. There are two types of data independence:

1. **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS today.
2. **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g., Adding or deleting attributes of a table should not affect the user's view of the table. But this type of independence is difficult to achieve as compared to physical data independence because some unavoidable changes in conceptual schema are reflected in the user's view. Logical data independence requires flexibility in the design

of the database, and the programmer has to foresee future requirements or modifications in the design. Both these conditions are difficult to meet completely in the practical world.

Phases of database design

Database designing for a real-world application starts from capturing the requirements to physical implementation using DBMS software which consists of the following steps shown below:

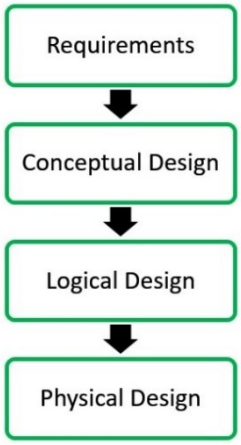


Figure 1.3.8: Database Design

Conceptual Design: The requirements of the database are captured using the high-level conceptual data model. For example, the ER model* is used for the conceptual design of the database.

Logical Design: Logical Design represents data in the form of a relational model. ER diagram* produced in the conceptual design phase is used to convert the data into the Relational Model.

Physical Design: In physical design, data in a relational model is implemented using commercial DBMS like Oracle, DB2.

NOTE: ER model and ER diagram will be covered in UNIT 2*

1.3.3 Database Application Architecture

A database application is a computer program whose primary purpose is entering and retrieving information from a computerized database.
Database applications allow users to gain access to the data they need. By using a database application, one can enter, update, read and delete data from a database.
Most end-user targeted database applications provide a **presentation layer** called **graphical user interface (GUI)** to work with. For example – Customer Relationship Management (CRM) applications, accounting applications, e-commerce applications running on World Wide Web.

There are mainly three types of database application architecture:

- Single-tier Architecture
- Two-tier Architecture
- Three-tier Architecture

Now, we will learn about the different architecture of database applications with diagrams.

Single-tier Architecture

Single-tier architecture is a centralized implementation of the database application. It is used for the development of the local application, where programmers can directly communicate with the database; here, both the **application (including both presentation layer and business logic layer)** and the **database system (DBMS)** physically reside at the client machine itself. The database system (DBMS) is invoked locally through query language statements thru the application. It does not need any network connection to perform actions on the database. For example – we develop a local application using MS-Access on our PC or laptop.

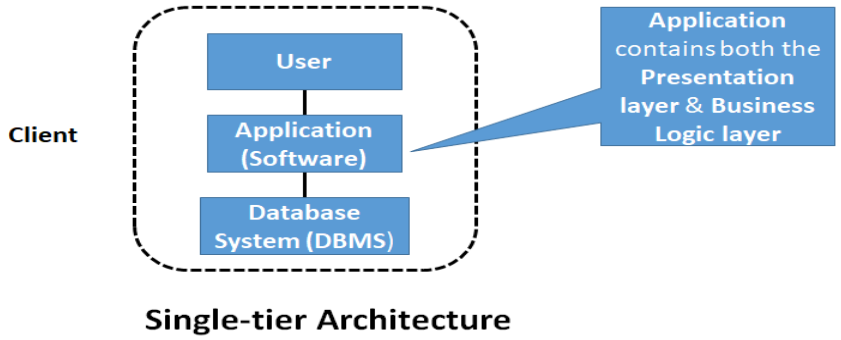


Figure 1. 13 : Single Tier Architecture

Two-tier Architecture

In a two-tier architecture, the **application (including both presentation layer and business logic layer)** resides at the client machine, where it invokes **database system (DBMS)** functionality, **over the network**, at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server. For example – desktop applications, games, music players, etc.

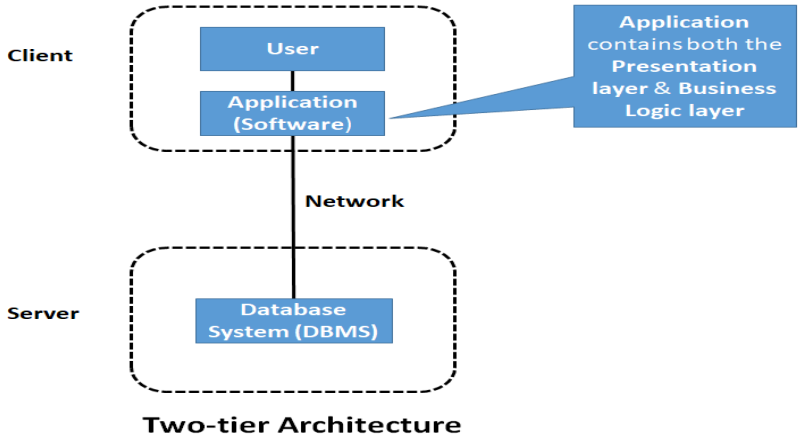
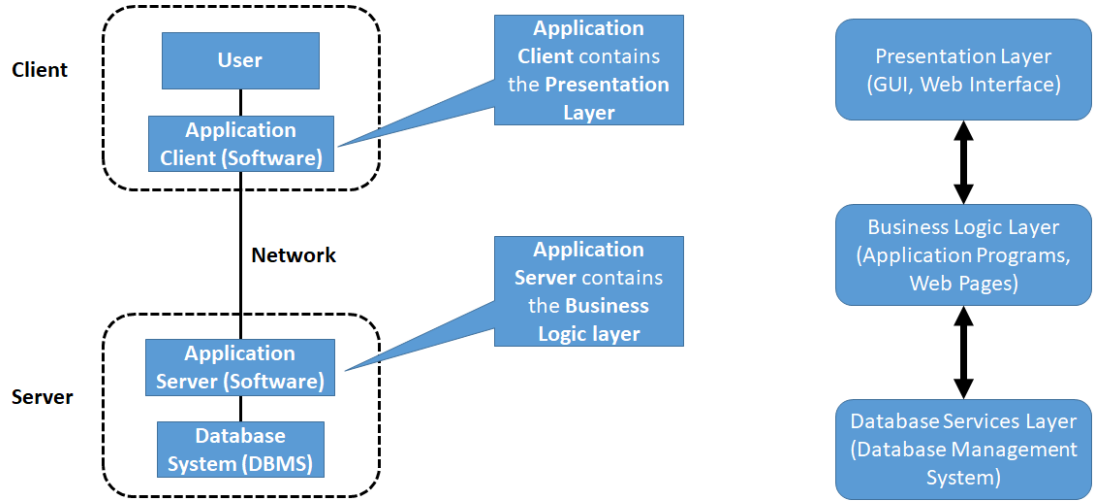


Figure 1. 14 : Two-Tier Architecture

Three-tier Architecture

In a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end (application client) communicates with an application server, usually through a presentation layer. The application server, in turn, communicates with a database system (DBMS) to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications and for applications that run on World Wide Web.



Three-tier Architecture

Figure 1. 15 : Three-Tier Architecture

1.3.4 Database Server Architecture

Database Server Architecture

Database server architecture defines the location where database management systems get managed by an organization. It depends upon the number of users, type of data access, and security requirements on the data.

A database management system can be used by only one person or group of people. It can be used on a single computer using a single application, or it can be used by multiple people using multiple devices like Desktop Applications, Web Applications, and Mobile Applications. Sometimes we have very critical data that need to be highly secure from destruction by people in the organization, hackers, or natural disasters.

Based on these requirements, the database administrator needs to decide the location of the database management system, known as database server architecture.
Database Server Architecture can be broadly categorized into three categories.

1. Centralized Database Server Architecture

2. Decentralized Database Server Architecture
3. Distributed Database Server Architecture

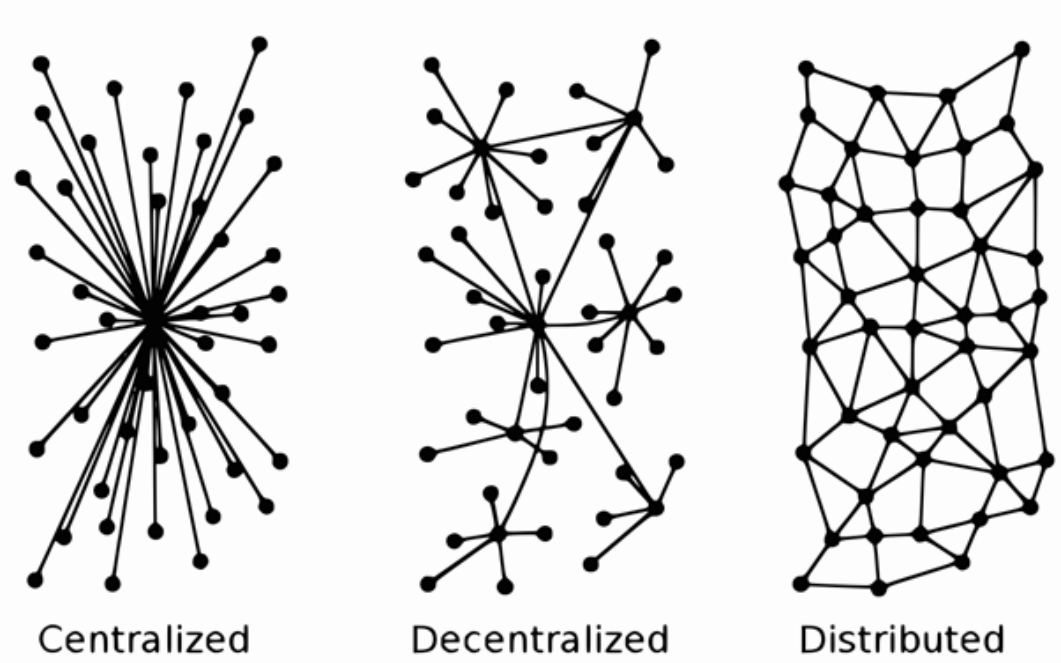


Figure 1. 16 : Types of Database Server Architecture

Centralized Database Server Architecture

In the case of centralized server architecture, the database management system is placed on a single server or centralized server. It is generally used when the organization is very small. This kind of database management requires fewer resources and costs, but it has some set of limitations. It requires regular backup. If we have not taken regular backup and something goes wrong with the server, then data can be lost.

All information sharing and communication is done via that central server. If that central server gets failed, then all the users get disconnected, and no information sharing is possible. It is also called the *central point of failure*.

Decentralized Database Server Architecture

In the case of decentralized server architecture, there are multiple servers, and each server has the same copy of the database. Different users can access data from different servers. If one server goes down due to some issue, then the users accessing that particular server will be affected, and the rest of the users will be working fine.

This server architecture also helps in load sharing among the users, and the performance of database management systems is not affected.

But there is always a limit to manage such servers, and such servers can be hacked or damaged. Data is controlled by database administrators who have permission to access and update the data.

Distributed Database

When the database is placed in such a manner that multiple copies of the same data are kept on multiple machines in a peer-to-peer model, it is known as distributed database server architecture. All such machines are known as nodes.

Once the data is updated on one machine, it gets automatically updated on pear nodes.
If data get corrupted on one node, it can be reinstalled back from another peer node. Since the data get placed on thousands of machines, it is considered to be non-hackable data.

For example, Cryptocurrencies like Bitcoin and Ethereum uses distributed database management. Data is managed on thousands of nodes. Anyone can make their own node to keep a local copy of the data. If you do any transaction using Bitcoin or Ethereum network, such data get updated on all the peer nodes connected in the network.

Even if your node is not online at a specific moment of time even then data will be updated automatically when you get connected with the network.

1.4 Applications of Database Management System

Applications, where we use Database Management System are:

- **Telecom**: There is a database to keep track of the information regarding calls made, network usage, customer details, etc. Without the database systems, it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Online shopping**: You must be aware of online shopping websites such as Amazon, Flipkart, etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.
- **Industry**: Where it is a manufacturing unit, warehouse, or distribution center, each one needs a database to keep the records of the ins and outs. For example distribution center should keep track of the product units that are supplied into the center as well as the products that got delivered out from the distribution center on each day; this is where DBMS comes into the picture.
- **Banking System**: Storing customer info, tracking day-to-day credit and debit transactions, generating bank statements, etc., has been done with the help of Database management systems.
- **Sales**: Store customer information, production information, and invoice details.

- **Airlines**: To travel through airlines, making early reservations; this reservation information, along with the flight schedule, is stored in a database.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details, etc.

1.5 Database Users

Database users, Database designers & Database administrators.

Database users: -

There are three main types of database-system users, differentiated by the way they expect to interact with the system.

- **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. Their main job function revolves around constantly querying and updating the database. These kinds of transactions are called **canned transactions**. For example, a clerk in the University who needs to add a new instructor to Department A invokes a program called NEW_HIRE. This program asks the clerk for the name of the new instructor, her new ID, the name of the Department (i.e., A), and the salary. The typical user interface for naive users is a forms interface, where the user can fill in appropriate fields of the form. Naive users may also simply read reports generated from the database.
- **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
- **Sophisticated users** interact with the system without writing application programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software. These include engineers, scientists, business analysts, and others who are thoroughly familiarized with the DBMS to meet their complex requirements.

Database Designer:

Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of the database designer to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

Database Administrator:

In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The function of a DBA include:

- **Schema definition**: The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access method definition**
- **Schema and physical organization modification**: The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization or to alter the physical organization to improve performance.
- **Granting of authorization for data access**: By granting the different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance**: Examples of database administrator's routine maintenance activities are:
    - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
    - Ensuring that enough free space is available for normal applications and upgrading disk space as required.
    - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

1.6 Review Questions

Multiple Choice Questions

Q.1 Immunity of the external schemas (or application programs) to changes in the conceptual schema is referred to as*: (ISRO CS 2018*)

a) Physical Data Independence
b) Logical Data Independence
c) Both (a) and (b)
d) None of the above

Q.2 The DBMS acts as an interface between what two components of an enterprise-class database system?

a) Database application and the database

b) Data and the database
c) The user and the database application
d) Database application and SQL

Q.3 Which of the following products was an early implementation of the relational model developed by E.F. Codd of IBM?
a) IDMS
b) DB2
c) dBase-II
d) R:base

Q.4 An application where only one user accesses the database at a given time is an example of_____.
a) single-user database application
b) multiuser database application
c) e-commerce database application
d) data mining database application

Q.5 The following are functions of a DBMS except _____.
a) creating and processing forms
b) creating databases
c) processing data
d) administrating databases

Q.6 DBMS should provide following feature(s) _____.
a) Protect data from system crash
b) All of these
c) Safety of the information stored
d) Authorized access

Q.7 A database is a complex type of _____.
a) None of these
b) Application
c) Data Structure
d) Manager

Q.8 Database is generally _____
a) System-centred
b) User-centred
c) Company-centred
d) Data-centred

Q.9 Which of the following isn't a level of abstraction?

a) physical
b) logical
c) user
d) view

Q.10 A level that describes how a record is stored.

a) physical
b) logical
c) user
d) view

Q.11 The _____level helps application programs hide the details of data types.

a) physical
b) logical
c) user
d) view

Q.12 A level that describes data stored in a database and the relationships among the data.

a) physical
b) logical
c) user
d) view

Q.13 There are numerous merits of using the database approach in contrast to file processing approaches. Which among the listed options is/are not true of using the database approach?

a) Data redundancy can be reduced.
b) Data inconsistency can be avoided to some extent.
c) Computing resources needed for data processing can be reduced
d) Data processing standards can be enforced.

Q.14 RX systems is a software company which is developing an application for a transport company. They have different teams involved in different modules of this project. Some are involved in database design; some are developing different front-end interfaces. One of these teams developing an interface using Java as the language. They are interacting with the database developed by other teams. Which of the user types are they falling under and how are they interacting with the database?

a) Naïve user, embedded SQL statements
b) Sophisticated user, query tools
c) Application programmer, embedded SQL statements
d) Parametric user, interface

Conceptual Questions

Q.1 Why would you choose a database system instead of simply storing data in operating system files? When would it make sense not to use a database system?

Q.2 What is data independence and how does a DBMS support it?

Q.3 Explain the difference between external, internal, and conceptual schemas. How are these different schema layers related to the concepts of logical and physical data independence?

Q.4 Define the roles and responsibilities of DBA.

Q.5 Differentiate 1-Tier, 2-Tier, 3-Tier architecture with diagram?

Q.6 What are the three types of data models?

Q.7 What are the disadvantages of data model?

Q.8 What are the advantages of data model?

Q.9 What are the limitations of hierarchical data model?

Q.10 Compare the network data model with hierarchical data model.

Q.11 Compare the network model with the relational model in terms of ease of learning and ease of use?

Q.12 What is the difference between a spreadsheet and a database? List three differences between them.

Q.13 What do you understand by Data Redundancy?

Project-Based Questions:

Q1. Suppose you want to build a video site similar to YouTube. Consider each of the points listed in Section 1.2 as disadvantages of keeping data in a file-processing system. Discuss the relevance of each of these points to the storage of actual video data and to metadata about the video, such as title, the user who uploaded it, tags, and which users viewed it.

Q2. Suppose the scenario of weather forecasting application, and we would like to store data, time, day, hours and forecasting status (rainy, dry, sunny, etc.). Discuss the following points

1. Types and structure of data we can use in this project.

2. Disadvantages in case of the file system.
3. Best Data model suitable for this scenario.