

# **Database Management System**

## **BIM SEM 4**

### **Unit 1: Introduction – Database Management Systems (LH 4)**

---

#### **Syllabus: -**

##### **Unit 1: Introduction – Database Management Systems (LH 4)**

Purpose of Database Systems. Data Abstraction. Data Models: The E-R Model, The Object-Oriented Model, The Relational Model, The Network Model, The Hierarchical Model, Physical Data Models. Instances and Schemes. Data Independence. Database Administrator. Database Users. Application Architecture (One tier, two tier and n-tier). Overall Database System Structure and Components.

---

## **Introduction**

### **Database**

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is a collection of data, typically describing the activities of one or more related organizations. 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 Systems**

A Database Management System (DBMS) is a software for creating and managing the databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. It stores data in such a way that it becomes easier to retrieve, manipulate, and update information.

Database Management System provides protection and security to the database. In the case of multiple users, it also maintains data consistency. Examples of popular DBMS are: MySQL, PostgreSQL, Access, Oracle, SQL Server, IBM, DB2 and Sybase.

#### **DBMS allows users the following tasks:**

- Data Definition: It is used for creation, modification, and removal 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.

## **Database Management System**

### **BIM SEM 4**

#### **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.

#### **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.
- Easy 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**

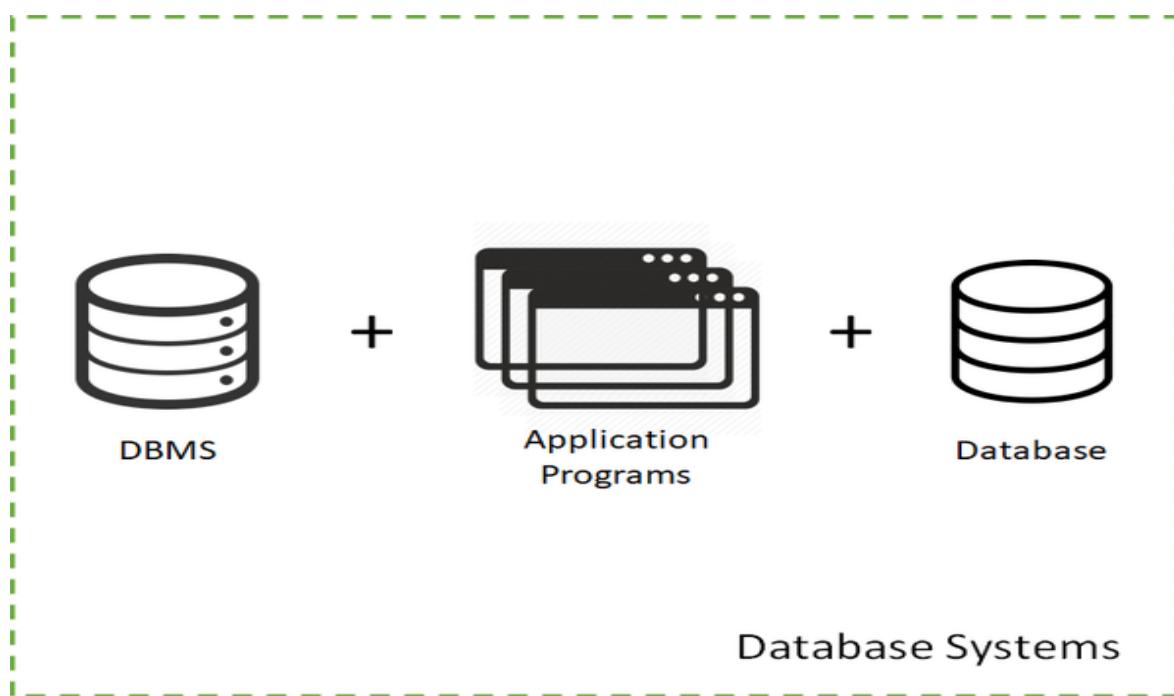
- Cost of Hardware and Software: It requires a high speed of data processor and large memory size to run DBMS software.
- Size: It occupies a large space of disks and large memory to run them efficiently.
- Complexity: Database system creates additional complexity and requirements.
- Higher impact of 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.

## **Database Management System**

### **BIM SEM 4**

#### **Database Systems.**

Database System is defined as a collection of application programs that interact with the database along with the DBMS and database Itself. It is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques.



#### **Purpose of Database Systems.**

Database systems arose in response to early methods of computerized management of commercial data. During the 1960th decade, A university used to record information about all instructors, students, departments and course offerings on a computer in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

- Add new students, instructors, and courses
- Register students for courses and generate class rosters
- Assign grades to students, compute grade point averages (GPA), and generate Transcripts

System programmers wrote these application programs to meet the needs of the university. New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science). As a result, the university used to create a new department and used to create new permanent files to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

## **Database Management System**

### **BIM SEM 4**

This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.

#### **Some characteristics of File Processing System are given below**

- It is a group of files storing data of an organization.
- Each file is independent from one another.
- Each file is called a flat file.
- Each file contained and processed information for one specific function, such as accounting or inventory.
- Files are designed by using programs written in programming languages such as COBOL, C, C++.
- The physical implementation and access procedures are written into database application; therefore, physical changes resulted in intensive rework on the part of the programmer.
- As systems became more complex, file processing systems offered little flexibility, presented many limitations, and were difficult to maintain.

Before the database management systems (DBMSs) were introduced, the organization usually used to store information in such systems. Hence, keeping organizational information in a file-processing system has a number of major disadvantages which are given below:

#### **Data redundancy and inconsistency.**

Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree.

#### **Data Mapping and Access:**

Although all the related information is grouped and stored in different files, there is no mapping between any two files. i.e.; any two dependent files are not linked. Even though Student files and Student\_Report files are related, they are two different files and they are not linked by any means. Hence if we need to display student details along with his report, we cannot directly pick from those two files. We have to write a lengthy program to search the Student file first, get all details, then go Student\_Report file and search for his report. When there is a very huge amount of data, it is always a time-consuming task to search for particular information from the file system. It is always an inefficient method to search for the data. Thus, the conventional file-processing environments do not retrieved data in a convenient and efficient manner.

#### **Data isolation:**

Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

## **Database Management System**

### **BIM SEM 4**

#### **Integrity problems:**

If we need to check for certain insertion criteria while entering the data into the file it is not possible directly. We can do it writing programs. Say, if we have to restrict the students above age 18, then it is by means of the program alone. There is no direct checking facility in the file system. Hence these kinds of integrity checks are not easy in the file systems.

#### **Atomicity problems:**

A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. i.e. if there is any failure to insert, update, or delete in the file system, there is no mechanism to switch back to the previous state.

Consider a program to transfer \$500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the \$500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic. It must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

Hence, atomicity refers to the completion of the whole transaction or not completing it at all. Partial completion of any transaction leads to incorrect data in the system. The file system does not guarantee atomicity. It may be possible with complex programs, but introduce for each transaction costs money.

#### **Concurrent-access anomalies:**

Accessing the same data from the same file is called concurrent access. In the file system, concurrent access leads to incorrect data. For example, a student wants to borrow a book from the library. He searches for the book in the library file and sees that only one copy is available. At the same time, another student also wants to borrow the same book and checks that one copy available. The first student opt for borrow and gets the book. But it is still not updated to zero-copy in the file and the second student also opt for borrow! But there are no books available. This is the problem of concurrent access to the file system.

#### **Security problems:**

The security of data is low in file-based system because, the data is maintained in the flat file(s) is easily accessible. For Example: Consider the Banking System. The Customer Transaction file has details about the total available balance of all customers. A Customer wants information about his account balance. In a file system it is difficult to give the Customer access to only his data in the file. Thus, enforcing security constraints for the entire file or for certain data items are difficult.

Overall, these problems and others led to the development of database management systems.

## **Database Management System**

### **BIM SEM 4**

#### **Advantage of DBMS over file system**

There are several advantages of Database management system over file system. Few of them are as follows:

- **Data redundancy and inconsistency:** -

Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining same files data for different applications. Hence changes made by one user does not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.

- **Data sharing:** -

File system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to centralized system.

- **Data concurrency:** -

Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user gets lost because of changes made by another user. File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.

- **Data searching:** -

For every search operation performed on file system, a different application program has to be written. While DBMS provides inbuilt searching operations. User only have to write a small query to retrieve data from database.

- **Data integrity:** -

There may be cases when some constraints need to be applied on the data before inserting it in database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user defined constraints on data by itself.

- **System crashing:** -

In some cases, systems might have crashes due to various reasons. It is a bane in case of file systems because once the system crashes, there will be no recovery of the data that's been lost. A DBMS will have the recovery manager which retrieves the data making it another advantage over file systems.

- **Data security:** -

A file system provides a password mechanism to protect the database but how longer can the password be protected? No one can guarantee that. This doesn't happen in the case of DBMS. DBMS has specialized features that help provide shielding to its data.

## **Database Management System**

### **BIM SEM 4**

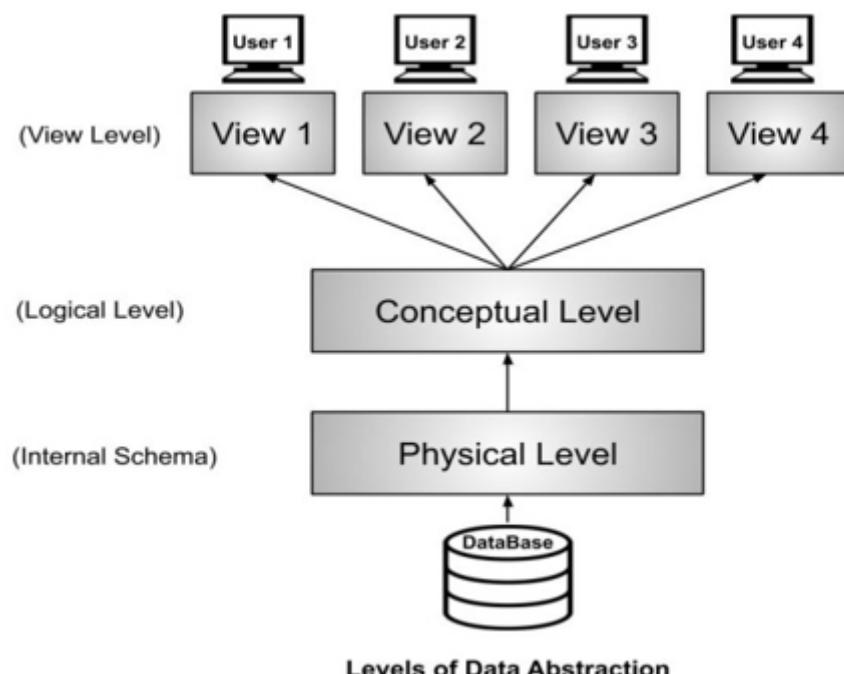
#### **Data Abstraction.**

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

There are three levels in data abstraction –

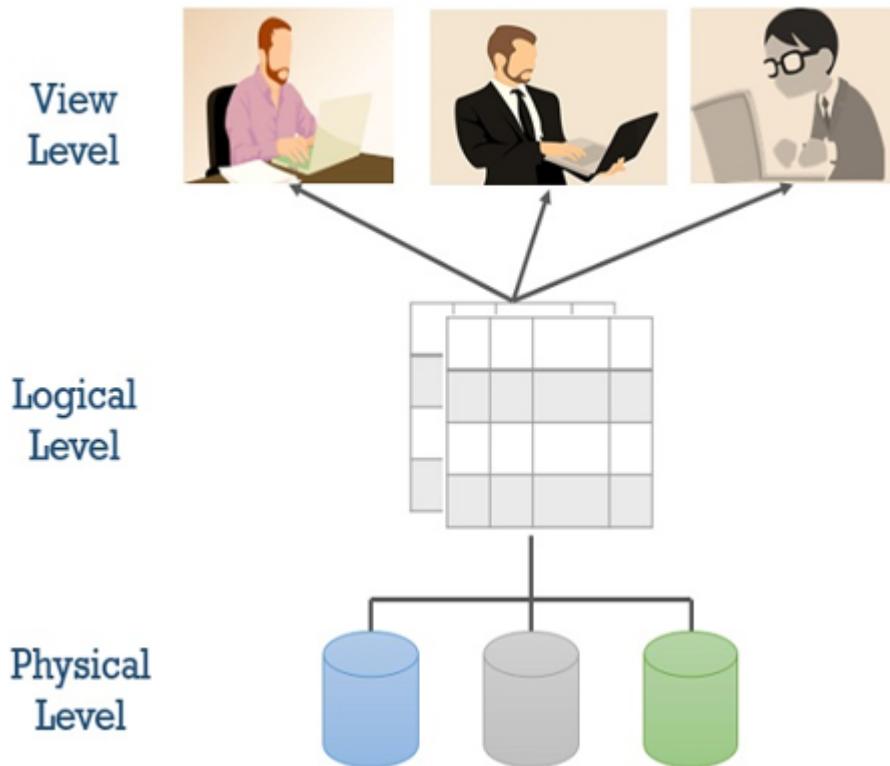
1. Physical level
2. Logical level
3. View level

The three levels in data abstraction is shown in figure below.



## Database Management System

### BIM SEM 4



#### Internal Level/Physical level:

This is the lowest level in data abstraction. This level describes how the data is actually stored in the physical memory like magnetic tapes, hard disks etc. In this level the file organization methods like hashing, sequential, B+ tree comes into picture. At this level, developer would know the requirement, size and accessing frequency of the records clearly. So, designing this level will not be much complex for him.

#### **Example:**

Consider we are storing customer information in a customer table. At physical level these records can be described as blocks of storage (bytes, gigabytes, etc.) in memory. These details are often hidden from the programmers.

#### **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

## **Database Management System**

### **BIM SEM 4**

#### **Logical level/ Conceptual level:**

This is the middle level of 3-level data abstraction architecture. It describes the actual data stored in the database in the form of tables and relates them by means of mapping. This level will not have any information on what a user views at external level. This level will have all the data in the database.

#### **Example:**

Consider we are storing customer information in a customer table.

At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

#### **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

#### **View level/ External level:**

This is the highest level in data abstraction. This level describes the user interaction with database system. At this level users see the data in the form of rows and columns. This level illustrates the users how the data is stored in terms of tables and relations. Users view full or partial data based on the business requirement. The users will have different views here, based on their levels of access rights. For example, student will not have access to see Lecturers salary details,

#### **Example**

Consider we are storing customer information in a customer table. At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

#### **Facts about external schema:**

- 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

## **Database Management System**

### **BIM SEM 4**

#### **Data Models:**

Data Model is a logical structure of Database. it defines how data will be stored, accessed and updated in a database management system and It also describes the design of database to reflect entities, attributes, relationship among data, constraints etc.

Data model is defined as an abstract model that organizes data description, data semantics and consistency constraints of data. Data model emphasizes on what data is needed and how it should be organized instead of what operations will be performed on data. Data Model is like architect's building plan which helps building conceptual models and set relationship between data items.

#### **Types of Data Models**

There are several types of data models in DBMS which are given below

#### **The E-R Model**

The Entity Relationship data model is based on the perception of the real world that consist of a collection of basics objects and relationships between them. Hence It is also called an object-based logical model and it is used for the conceptual design of a database.

While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

The Entity Relationship data model is high-level data model. And was developed by Chen in 1976. It is very simple and easy to design logical view of data. The developer can easily understand the system by looking at an ER model constructed.

ER Model is based on –

- Entities and their attributes.
- Relationships among entities.

These concepts are explained below.

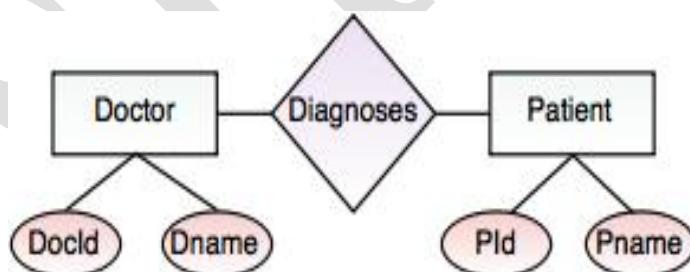


Fig. ER Model

## **Database Management System**

### **BIM SEM 4**

In this diagram,

- **Rectangle** represents the entities. An entity in an ER Model is a real-world entity having properties called attributes. It can be a person, place, or even a concept. For example, in above diagram, Doctor and Patient are considered as an entity.
- **Ellipse** represents the **attributes**. An entity contains a real-world property called attribute. Every attribute is defined by its set of values called domain. For example, in above diagram, the entity Doctor have **attributes** like Docid, Dname and entity patient have attribute Pid, Pname. Attribute describes each entity becomes a major part of the data stored in the database.
- **Diamond** represents the **Relationship** in ER diagrams. The logical association among entities is called relationship. Eg. Doctor diagnoses the Patient.
- The **Relationships** are mapped with entities in various ways. Mapping cardinalities define the number of associations between two entities. Mapping cardinalities are –
  - one to one
  - one to many
  - many to one
  - many to many

### **Advantages of ER Model**

- **Simple:** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities, we can easily build the ER Diagram for the model.
- **Effective Communication Tool:** This model is used widely by the database designers for communicating their ideas.
- **Easy Conversion to any Model:** This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

### **Disadvantages of ER Model**

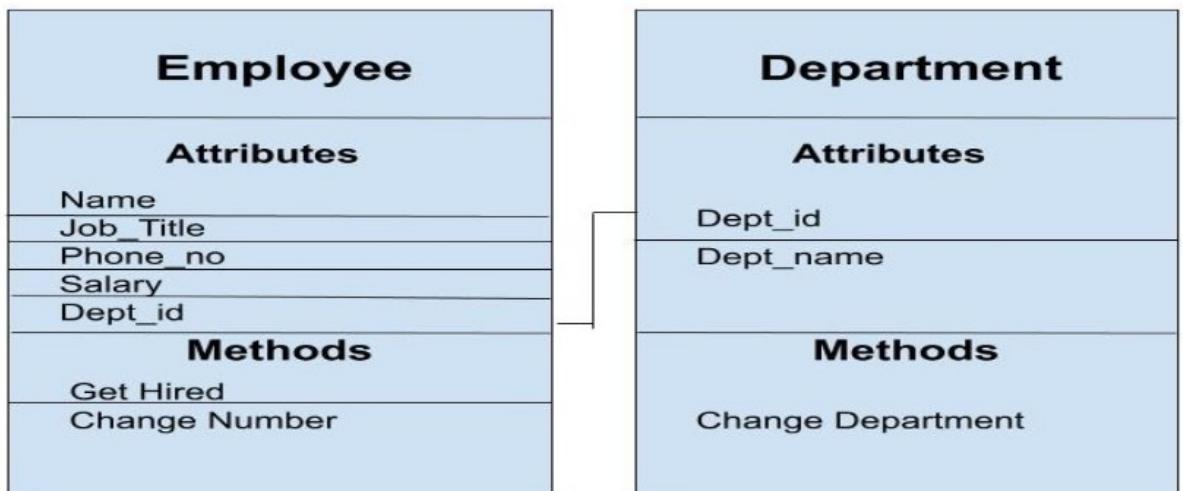
- **No industry standard for notation:** There is no industry standard for developing an ER model. So, one developer might use notations which are not understood by other developers.
- **Hidden information:** Some information might be lost or hidden in the ER model. As it is a high-level view so there are chances that some details of information might be hidden.

## **Database Management System**

### **BIM SEM 4**

#### **The Object-Oriented Model**

- The real-world problems are more closely represented through the object-oriented data model. In this model, both the data and relationship are present in a single structure known as an object.
- Object model stores the data in the form of objects, classes and inheritance. And It is easy to maintain and modify the existing code.
- This model handles more complex applications, such as Geographic Information System (GIS), scientific experiments, engineering design and manufacturing.
- We can store audio, video, images, etc in the database which was not possible in the relational model. In this model, two or more objects are connected through links. We use this link to relate one object to other objects. This can be understood by the example given below.



#### **Object\_Oriented\_Model**

- In the above example, we have two objects Employee and Department. All the data and relationships of each object are contained as a single unit. The attributes like Name, Job\_title of the employee and the methods which will be performed by that object are stored as a single object. The two objects are connected through a common attribute i.e the Department\_id and the communication between these two will be done with the help of this common id.

## Database Management System

### BIM SEM 4

#### The Relational Model

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns.

Each column lists an attribute of the entity. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, where it's called a foreign key.

The basic structure of a relational model is tables. So, the tables are also called relations in the relational model.

Example: Employee table.

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

**EMPLOYEE TABLE**

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model around the world. In relational model, the data and relationships are represented by collection of inter-related tables.

#### Features of Relational Model

- **Tuples:** Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
- **Attribute or field:** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above employee table, we have different attributes of the employee like Salary, Mobile\_no, etc.

#### Advantages of Relational Model

- **Simple:** This model is simpler as compared to the network and hierarchical model.
- **Scalable:** This model can be easily scaled as we can add as many rows and columns we want. The relational data model makes it easy to design, implement, maintain, uses the database.
- **Structural Independence:** We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data, we can say that structural independence has been achieved.

## Database Management System

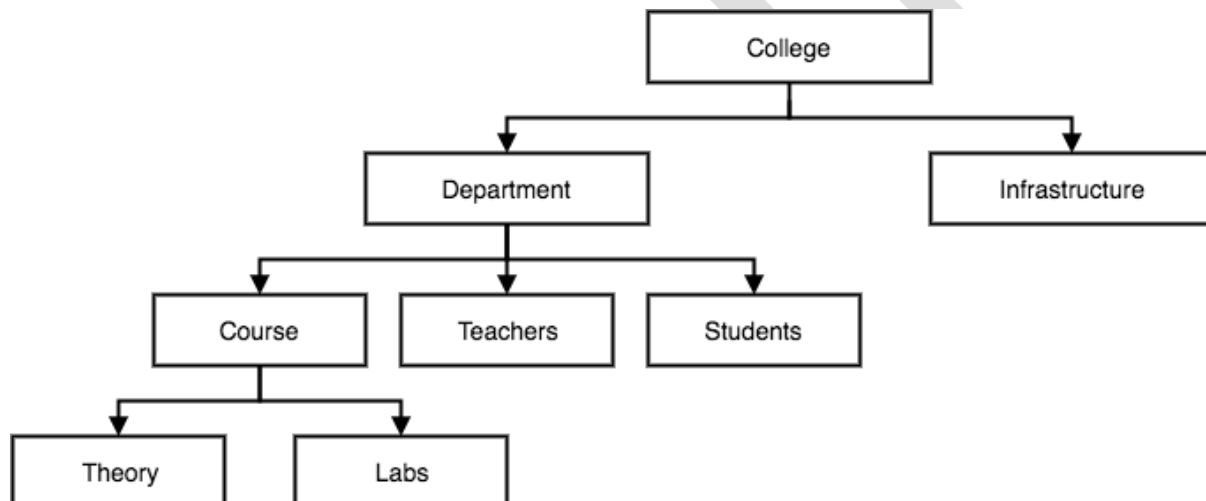
### BIM SEM 4

#### Disadvantages of Relational Model

- **Hardware Overheads:** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.
- **Bad Design:** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows.

#### The Hierarchical Model

- Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure with a single root, to which all the other data is linked. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node.
- **The hierarchical model is based on the parent-child hierarchical relationship. In this model, there is one parent entity with several children entity.**
- This model easily represents some of the real-world relationships like index of a book, food recipes, sitemap of a website etc.
- **Example:** -In the following figure , at the top, there should be only one entity which is called root. collage is the parent entity called root and it has several children entities like department, infrastructure and many more.



**Figure: - Hierarchical Model**

#### Features of a Hierarchical Model

- **One-to-many relationship:** The data here is organised in a tree-like structure where the one-to-many relationship is between the datatypes. Also, there can be only one path from parent to any node.  
**Example:** In the above example, if we want to go to the node theory, we only have one path to reach there i.e through department to course to theory.
- **Parent-Child Relationship:** Each child node has a parent node but a parent node can have more than one child node. **Multiple parents are not allowed.**

## **Database Management System**

### **BIM SEM 4**

- **Deletion Problem:** If a parent node is deleted then the child node is automatically deleted.
- **Pointers:** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. Example: In the above example the 'collage' node points to the two other nodes 'department' node and 'infrastructure' node.

#### **Advantages of Hierarchical Model**

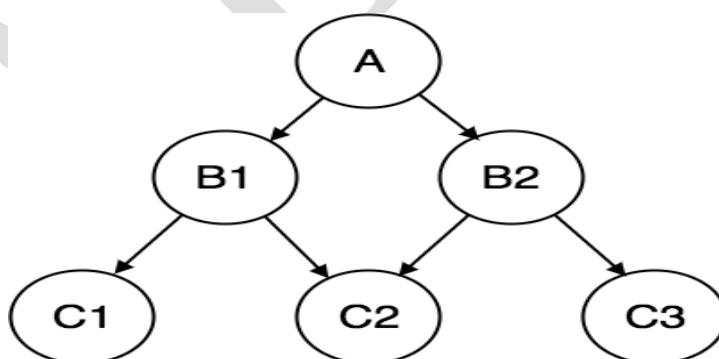
- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

#### **Disadvantages of Hierarchical Model**

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent nodes then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted.

#### **The Network Model**

- This model is an extension of the hierarchical model. It was the most popular model before the relational model.
- In the network data model, data are represented by collections of records. Relationships among data are represented by links.
- In this database model, data is more related as more relationships are established. Also, as the data is more related, the data accessing is also easier and fast. This database model was used to map many-to-many data relationships.
- This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph.
- **Example:** In the example below, we can see that node C2 has two parents i.e. B1 and B2. This was earlier not possible in the hierarchical model.



**Figure: - Network Model**

# Database Management System

## BIM SEM 4

### Features of a Network Model

- **Ability to Merge more Relationships:** In this model, as there are more relationships so data is more related. This model has the ability to manage one-to-one relationships as well as many-to-many relationships.
- **Many paths:** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
- **Circular Linked List:** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

### Advantages of Network Model

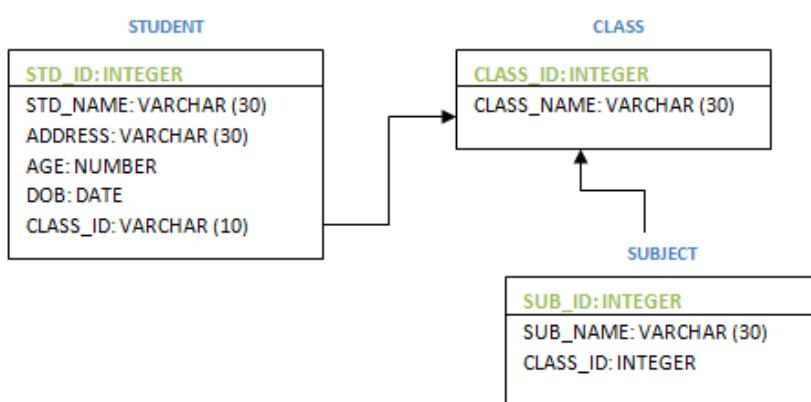
- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So, the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

### Disadvantages of Network Model

- As more and more relationships need to be handled the system will be complex. So, a user must be having detailed knowledge of the model to work with the model.
- Any change like updation, deletion, insertion is very complex.

## Physical Data Models

- Physical data model represents the model where it describes how data are stored in computer memory, how they are scattered and ordered in the memory, and how they would be retrieved from memory.
- Basically, physical data model represents the data at data layer or internal layer. It represents each table, their columns and specifications, constraints like primary key, foreign key etc. It basically represents how each table are built and related to each other in DB.



- Above diagram shows how physical data model is designed. It is represented as UML diagram along with table and its columns. Primary key is represented at the top. The relationship between the tables is represented by interconnected arrows from table to

## **Database Management System**

### **BIM SEM 4**

table. Above STUDENT table is related to CLASS and SUBJECT is related to CLASS. The above diagram depicts CLASS as the parent table and it has 2 child tables – STUDENT and SUBJECT.

**In short, we can say a physical data model has**

- Tables and its specifications – table names and their columns. Columns are represented along with their datatypes and size. In addition, primary key of each table is shown at the top of the column list.
- Foreign keys are used to represent the relationship between the tables. Mapping between the tables are represented using arrows between them.
- Physical data model can have denormalized structure based on the user requirement. The tables might not be in normalized forms.

**Note:** -Physical data model is dependent on the RDBMS i.e.; it varies based on the RDBMS used. This means datatype notation varies depending on the RDBMS.

### **Instances and Schemas.**

#### **DBMS Instance**

- The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.
- An instance is the information collected in a database at some specific moment, and it is also known as state or extension. It is a snapshot where the current state or occurrence of a database is framed at that moment. Each time when the data is inserted or deleted from the database changes the state of the database that is the reason why an instance of the database changes very often.
- The schema of the database is specified to the DBMS when a new database is defined, at that time the corresponding database is empty, hence has an empty instance. The starting state of the database is acquired when the database is first loaded with initial data. From then onwards, each time the data is updated we get a new database instance. At any point in time, there is a current state associated with a database. The DBMS is partially accountable for confirming the valid instance of a database where the instance assures the structure and constraints specified in the schema.
- Let's take the similar example in the instance. Here the student construct will contain their individual entities in the attributes.

## **Database Management System**

### **BIM SEM 4**

#### **STUDENT**

Name	Student_number	Semester
Antony	1021	1
Bob	1022	1
Deeksha	1023	1
Rohan	1024	1

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

#### **Database Schema**

- The overall design of a database is called the database schema. In other words, schema is an overall structure of a database. Schema defines how database is stored, relationship between entities and attributes and so on. Database systems have several schemas, partitioned according to the level of abstraction.
- It is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.
- A schema diagram, as shown below, displays only names of record types (entities) and names of data items (attributes) and does not show the relationships among the various files.

## Database Management System

### BIM SEM 4

P	P#	PNAME	PRODUCT	PRICE	CITY
C	C#	CNAME	ADD	CITY	
PC	P#	C#	CITY		
<hr/>					
P#	PNAME	PRODUCT	PRICE	CITY	
P1	Rahat Computers	Printer	5000	Patiala	
P2	Ruhani info system	Monitor	6000	Jalandhar	
P3	IBM	Keyboard	1200	Qadian	

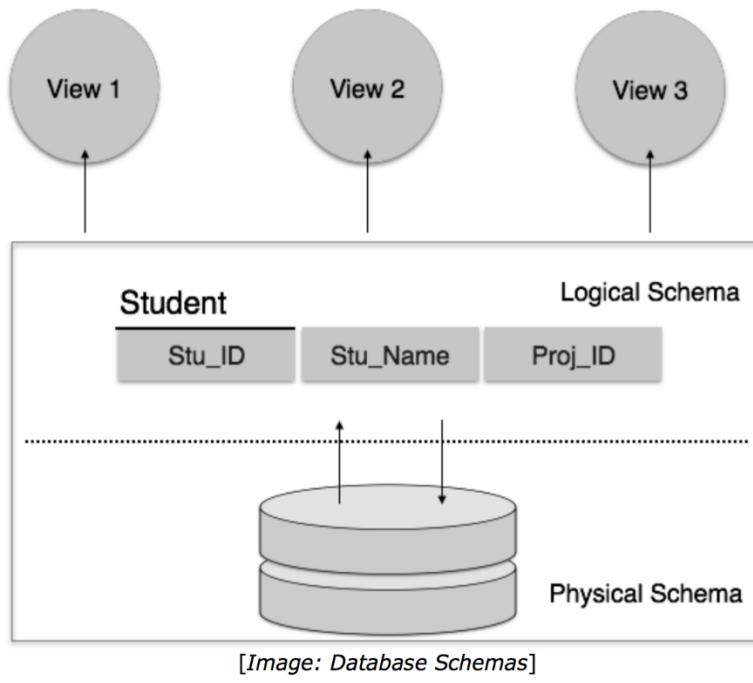
- The schema will remain the same while the values filled into it change from instant to instant. When the schema framework is filled in with data item values, it is referred as an instance of the schema. The data in the database at a particular moment of time is called a database state or snapshot, which is also called the current set of occurrences or instances in the database

A database schema can be divided broadly into three categories:

- **Physical Schema:** This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Schema:** This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.
- **View schema:** View schema can be defined as the design of the database at the view level, which generally describes end-user interaction with database systems.

## Database Management System

### BIM SEM 4



**For example:** Let suppose you are storing students' information on a student's table. At the physical level, these records are described as chunks of storage (in bytes, gigabytes, terabytes, or higher) in memory, and these elements often remain hidden from the programmers. Then comes the logical level; here at a logical level, these records can be illustrated as fields and attributes along with their data type(s); their relationship with each other can be logically implemented. Programmers generally work at this level because they are aware of such things about database systems. At view level, a user can able to interact with the system, with the help of GUI, and enter the details on the screen. The users are not aware of the fact of how the data is stored and what data is stored; such features are hidden from them.

#### Difference between Schema and Instance:

SCHEMA	INSTANCE
<ul style="list-style-type: none"><li>It is the overall description of the database.</li></ul>	<ul style="list-style-type: none"><li>It is the collection of information stored in a database at a particular moment.</li></ul>
<ul style="list-style-type: none"><li>Schema is same for whole database.</li></ul>	<ul style="list-style-type: none"><li>Data in instances can be changed using addition, deletion, updation.</li></ul>
<ul style="list-style-type: none"><li>Does not change Frequently.</li></ul>	<ul style="list-style-type: none"><li>Changes Frequently.</li></ul>
<ul style="list-style-type: none"><li>Defines the basic structure of the database i.e how the data will be stored in the database.</li></ul>	<ul style="list-style-type: none"><li>It is the set of Information stored at a particular time.</li></ul>

## **Database Management System**

### **BIM SEM 4**

#### **Data Independence.**

It can be defined as the ability of the data in one schema not to be affected whenever there is an update or modify operation performed on the other schema in the database system.

It is usually applied to the three-tiered Database Architecture systems, where the data from the bottom tier are expected to be not distressed in the events of modifications performed on the other tiers on the top of the system or it is the capacity to modify a scheme definition in one level without affecting a scheme definition in a higher level.

The purposes of instilling Data Independence on to the Database Management System are for maintaining the functional behaviour of the database system, enhanced security of the system, keeping the boundaries clear between the three levels of the database schema, performing application maintenance without the extra cost being spent, etc.

**Data Independence can be classified into two different types, with respect to the levels of the database systems. They are**

- 1. Physical Data Independence**
- 2. Logical Data Independence**

#### **1. Physical data independence:**

It is the capacity to modify the physical scheme without having alteration to the conceptual schema.

For example, a change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

#### **2. Logical data independence:**

It is the capacity to modify the logical or conceptual scheme (user view) without having alteration in external schema.

For example, the addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

#### **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
- Easily make modifications in the physical level is needed to improve the performance of the system.

## **Database Management System**

### **BIM SEM 4**

#### **Database Administrator**

Database administrator (DBA) is a person controlling and coordinating the database management system and responsible for authorising access to database, coordinating and monitoring its use and for acquiring software and hardware resources as needed.

A DBA in DBMS works on creating, maintaining, querying, and tuning the database of the organization. This role requires the professionals to have good knowledge and experience in the particular RDBMS that the company uses. Based on the requirements of the company, there are various types of DBAs including:

- **Administrative DBA:** They maintain and run the databases and servers of the organization. They are mainly concerned with the Data security, replication, and backup of data.
- **Development DBA:** They work on developing SQL queries and stored procedures to meet the requirements of the business. They specialize in database development.
- **Data Architect:** They design schemas, build data structures, table indexes, and relationships. They are mainly responsible for building a structure that meets the business requirements in a specific area.
- **Data Warehouse DBA:** They merge data from numerous data sources and store them in a data warehouse.

#### **Responsibilities of a Database Administrator**

A database administrator's primary job is to ensure that data is available, protected from loss and corruption, and easily accessible as needed. Below are some of the responsibilities that make up the day-to-day work of a DBA.

##### **1. Software installation and Maintenance**

A DBA often collaborates on the initial installation and configuration of a new Oracle , SQL Server database etc. The system administrator sets up hardware and deploys the operating system for the database server, then the DBA installs the database software and configures it for use. As updates and patches are required, the DBA handles this on-going maintenance. And if a new server is needed, the DBA handles the transfer of data from the existing system to the new platform.

##### **2. Data Extraction, Transformation, and Loading**

Known as ETL, data extraction, transformation, and loading refer to efficiently importing large volumes of data that have been extracted from multiple systems into a data warehouse environment. This external data is cleaned up and transformed to fit the desired format so that it can be imported into a central repository.

##### **3. Specialised Data Handling**

Databases can be massive and may contain unstructured data types such as images, documents, or sound and video files. Managing a very large database (VLDB) may require higher-level skills and additional monitoring and tuning to maintain efficiency.

# **Database Management System**

## **BIM SEM 4**

### **4. Database Backup and Recovery**

DBAs create backup and recovery plans and procedures based on industry best practices, then make sure that the necessary steps are followed. Backups takes high cost, time and money, so the DBA may have to persuade management to take necessary precautions to preserve data.

System admins or other personnel may actually create the backups, but it is the DBA's responsibility to make sure that everything is done on schedule.

### **5. Security**

A DBA needs to know potential weaknesses of the database software and the company's overall system and work to minimise risks. No system is one hundred per cent immune to attacks, but implementing best practices can minimise risks. In the case of a security breach or irregularity, the DBA can consult audit logs to see who has done what to the data. Audit trails are also important when working with regulated data.

### **6. Authentication**

Setting up employee access is an important aspect of database security. DBAs control who has access and what type of access they are allowed. For instance, a user may have permission to see only certain pieces of information, or they may be denied the ability to make changes to the system.

### **7. Capacity Planning**

The DBA needs to know how large the database currently is and how fast it is growing in order to make predictions about future needs. Storage refers to how much room the database takes up in server and backup space. Capacity refers to usage level. If the company is growing quickly and adding many new users, the DBA will have to create the capacity to handle the extra workload.

### **8. Performance Monitoring**

Monitoring databases for performance issues is part of the on-going system maintenance a DBA performs. If some part of the system is slowing down processing, the DBA may need to make configuration changes to the software or add additional hardware capacity. Many types of monitoring tools are available, and part of the DBA's job is to understand what they need to track to improve the system.

### **9. Database Tuning**

Performance monitoring shows where the database should be tweaked to operate as efficiently as possible. The physical configuration, the way the database is indexed, and how queries are handled can all have a dramatic effect on database performance. With effective monitoring, it is possible to proactively tune a system based on application and usage instead of waiting until a problem develops.

### **10. Troubleshooting**

DBAs are on call for troubleshooting in case of any problems. Whether they need to quickly restore lost data or correct an issue to minimise damage, a DBA needs to quickly understand and respond to problems when they occur.

## **Database Management System**

### **BIM SEM 4**

## **Database Users**

Database users are the persons who interact with the database and take the benefits of database. They can be programmers, scientist, engineers, business person or can be an employee.

They are differentiated into different types based on their need and way of accessing the database.

1. **Application Programmer** – Application programmer are computer professional, who writes application program. They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc.

They develop application programs and provide user interface through application so the user can interact with the database.

For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.

2. **Sophisticated Users** – They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL.

These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.

3. **Specialized Users** – These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.

These may be Computer aided design and drafting (CADD) systems, knowledge-based and expert systems, complex data systems (audio/video), etc.

4. **Stand-alone Users** – These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces.

5. **Native Users** – these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfil their requests.

## **Database Management System**

### **BIM SEM 4**

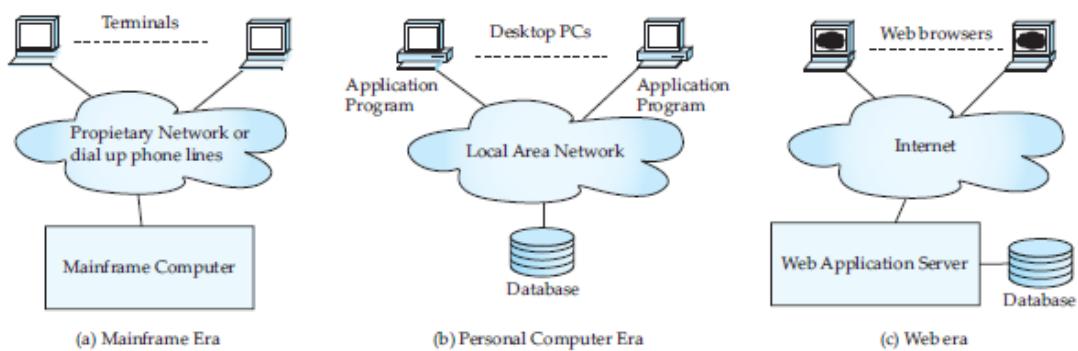
#### **Application Architecture (One tier, two tier and n-tier).**

Although many people interact with databases, very few people use a query language to interact with a database system directly. The most common way in which users interact with databases is through an application program that provides a user interface at the front end, and interfaces with a database at the back end. Such applications take input from users, typically through a forms-based interface, and either enter data into a database or extract information from a database based on the user input, and generate output, which is displayed to the user.

As an example of an application, consider a university registration system. Like other such applications, the registration system first requires you to identify and authenticate yourself, typically by a user name and password. The application then uses your identity to extract information, such as your name and the courses for which you have registered, from the database and displays the information. The application provides a number of interfaces that let you register for courses and query a variety of other information such as course and instructor information. Organizations use such applications to automate a variety of tasks, such as sales, purchases, accounting and payroll, human-resources management, and inventory management, among many others.

A typical application program includes a front-end component, which deals with the user interface, a back-end component, which communicates with a database, and a middle layer, which contains “business logic,” that is, code that executes specific requests for information or updates, enforcing rules of business such as what actions should be carried out to execute a given task, or who can carry out what task.

#### **Application architectures in different eras**



**Figure 9.1 Application architectures in different eras**

Application architectures have evolved over time, as illustrated in Figure above. Applications such as airline reservations have been around since the 1960s. In the early days of computer applications, applications ran on a large “mainframe” computer, and users interacted with the application through terminals, some of which even supported forms.

With the widespread use of personal computers, many organizations used a different architecture for internal applications, with applications running on the user’s computer, and

## **Database Management System**

### **BIM SEM 4**

accessing a central database. This architecture, often called a “client–server” architecture, allowed the creation of powerful graphical user interfaces, which earlier terminal-based applications did not support. However, software had to be installed on each user’s machine to run an application, making tasks such as upgrades harder. Even in the personal computer era, when client–server architectures became popular, mainframe architecture continued to be the number of geographically distributed locations.

In the recent years, Web browsers have become the universal front end to database applications, connecting to the back end through the Internet. Browsers use a standardized syntax, the HyperText Markup Language (HTML) standard, which supports both formatted display of information, and creation of forms-based interfaces. The HTML standard is independent of the operating system or browser, and pretty much every computer today has a Web browser installed. Thus, a Web-based application can be accessed from any computer that is connected to the Internet.

Unlike client–server architectures, there is no need to install any application specific software on client machines in order to use Web-based applications. However, sophisticated user interfaces, supporting features well beyond what is possible using plain HTML, are now widely used, and are built with the scripting language JavaScript, which is supported by most Web browsers. JavaScript programs, unlike programs written in C, can be run in a safe mode, guaranteeing they cannot cause security problems. JavaScript programs are downloaded transparently to the browser and do not need any explicit software installation on the user’s computer.

### **Types of Application architecture/database Architecture**

Database architecture uses programming languages to design a particular type of software for businesses or organizations. Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users.

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. The tiers are classified as follows:

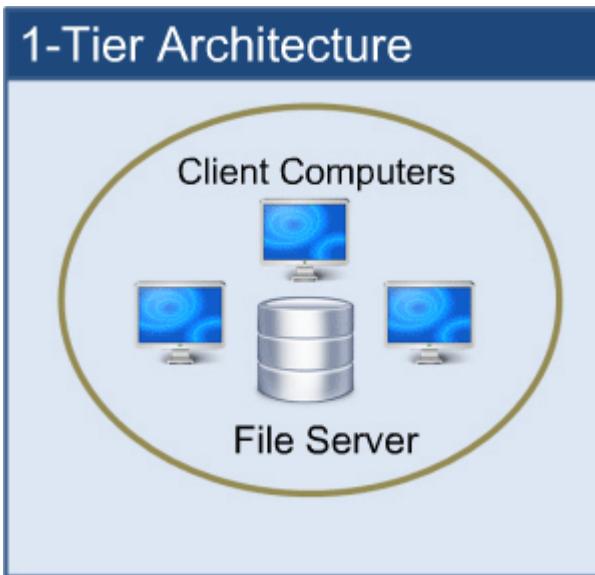
- 1-tier architecture
- 2-tier architecture
- 3-tier architecture
- n-tier architecture

#### **1-tier architecture:**

One-tier architecture involves putting all of the required components for a software application or technology on a single server or platform.

## Database Management System

### BIM SEM 4



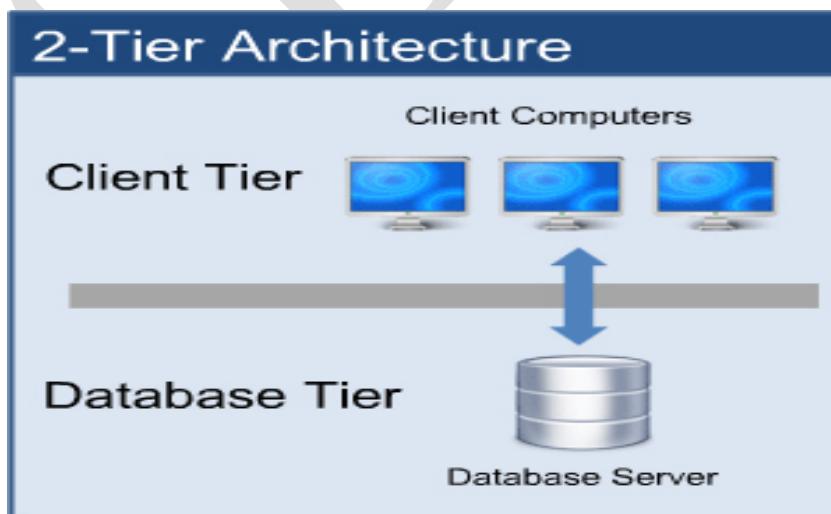
1-tier architecture

Basically, a one-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place. Developers see these types of systems as the simplest and most direct way. So, it can be concluded as follows

- 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:

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.



2-tier architecture

# Database Management System

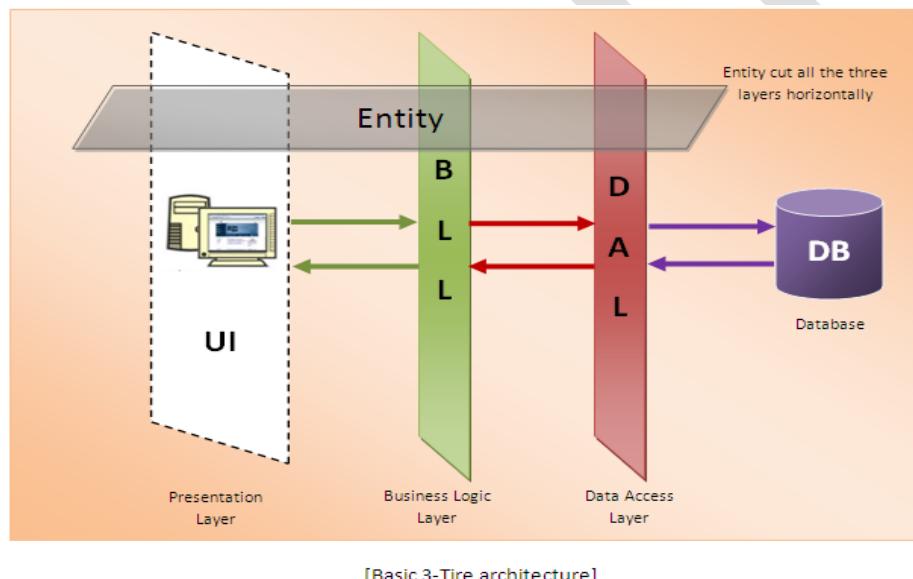
## BIM SEM 4

More about 2-Tier architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, 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 architecture:

- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.
- 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.



This architecture has different usages with different applications. It can be used in web applications and distributed applications. It consists of.

**Database (Data) Tier:** - At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

**Application (Middle) Tier:** - At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At

## **Database Management System**

### **BIM SEM 4**

the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

**User (Presentation) Tier:** - End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

#### **n-tier architecture:**

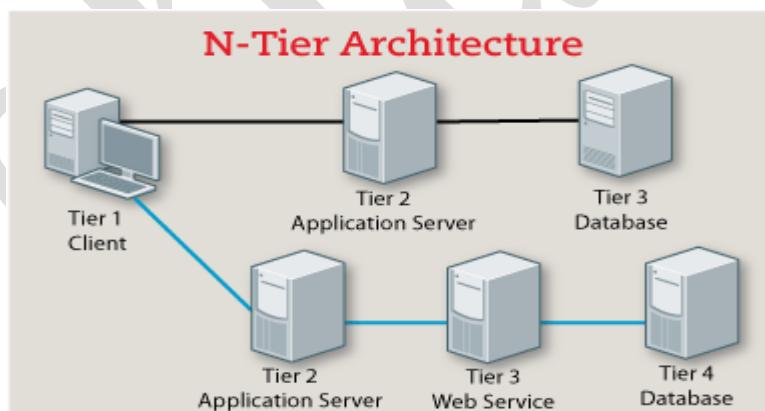
N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated. That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity. The “N” in the name n-tier architecture refers to any number from 1.

It is the physical separation of the different parts of the application as opposed to the usually conceptual or logical separation of the elements in the model-view-controller (MVC) framework.

In MVC, there is no actual middle layer because the interaction is triangular; the control layer has access to both the view and model layers and the model also accesses the view; the controller also creates a model based on the requirements and pushes this to the view.

N-tier architecture would involve dividing an application into three different tiers. These would be the

- logic tier,
- the presentation tier, and
- the data tier.



**Fig:- n-tier architecture**

#### **Presentation tier**

The top-most level of the application is the user interface. the main function of the interface is to translate tasks and results to something the user can understand.

#### **Logic tier**

## **Database Management System**

### **BIM SEM 4**

This layer coordinates the application, processed commands, makes logical decision and evaluations, and performs calculations.it also moves and processes data between the two surrounding layers.

#### **Data tier**

In this tier the information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing , and then eventually back to user.

#### **Benefits of N-Tier Architecture**

There are several benefits to using n-tier architecture for your software. These are scalability, ease of management, flexibility, and security.

- **Secure:** You can secure each of the three tiers separately using different methods.
- **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
- **Flexible:** Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.

#### **Overall Database System Structure and Components.**

DBMS (Database Management System) acts as an interface between the user and the database. The user requests; the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

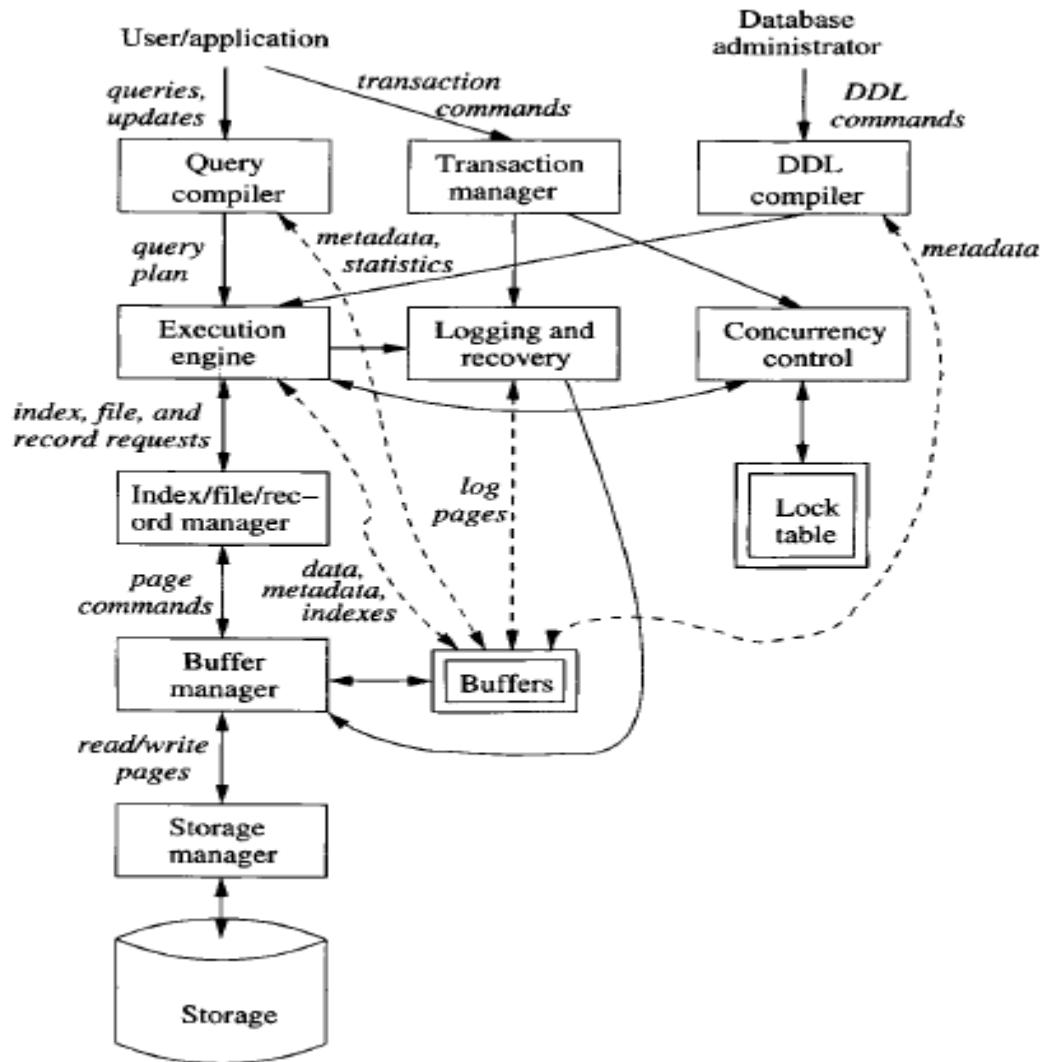
In Figure below shows an outline of a complete DBMS. Single boxes represent system components, while double boxes represent in-memory data structures. The solid lines indicate control and data flow, while dashed lines indicate data flow only.

First, at the top, we suggest that there are two **distinct sources of commands** to the DBMS:

1. **Conventional users and application programs** that ask for data or modify data.
2. **A database administrator:** a person or persons responsible for the structure or schema of the database.

# Database Management System

## BIM SEM 4



**Figure: Database management system components**

### Data-Definition Language Commands

This command is shown at the upper right side of Fig. above. The DBA, who needs special authority to execute schema-altering uses this command. The data-definition language (DDL) commands are parsed by a DDL processor and passed to the execution engine, which then goes through the index/file/record manager to alter the metadata, that is, the schema information for the database.

### Overview of Query Processing

A user or an application program initiates some action, using the data-manipulation language (DML) which is shown on the left side of Fig above. This command does not affect the schema of the database, but may affect the content of the database or will extract data from the database. DML statements are handled by two separate subsystems, as follows.

#### **1.Answering the Query**

The query is parsed and optimized by a **query compiler**. The resulting **query plan**, or sequence of actions the DBMS will perform to answer the query, is passed to the

## **Database Management System**

### **BIM SEM 4**

**execution engine.** The execution engine issues a sequence of requests for small pieces of data, typically records or tuples of a relation, to a **resource manager** that knows about data files (holding relations), the format and size of records in those files, and index files, which help find elements of data files quickly. The requests for data are passed to the **buffer manager**. The **buffer manager**'s task is to bring appropriate portions of the data from secondary storage (disk) where it is kept permanently, to the main-memory buffers. Normally, the page or “disk block” is the unit of transfer between buffers and disk. The buffer manager communicates with a **storage manager** to get data from disk. The storage manager might involve operating-system commands, but more typically, the DBMS issues commands directly to the disk controller.

#### **The Component of Query Processor**

The portion of the DBMS that most affects the performance that the user sees is the query processor. In Fig. above the query processor is represented by two components:

**I. The query compiler**, which translates the query into an internal form called a query plan. The latter is a sequence of operations to be performed on the data. Often the operations in a query plan are implementations of “relational algebra” operations. The query compiler consists of three major units:

**(a) A query parser**, which builds a tree structure from the textual form of the query.

**(b) A query pre-processor**, which performs semantic checks on the query (e.g., making sure all relations mentioned by the query actually exist), and performing some tree transformations to turn the parse tree into a tree of algebraic operators representing the initial query plan.

**(c) A query optimizer**, which transforms the initial query plan into the best available sequence of operations on the actual data. The query compiler uses metadata and statistics about the data to decide which sequence of operations is likely to be the fastest. For example, the existence of an index, which is a specialized data structure that facilitates access to data, given values for one or more components of that data, can make one plan much faster than another.

**II. The execution engine**, which has the responsibility for executing each of the steps in the chosen query plan. The execution engine interacts with most of the other components of the DBMS, either directly or through the buffers. It must get the data from the database into buffers in order to manipulate that data. It needs to interact with the scheduler to avoid accessing data that is locked, and with the log manager to make sure that all database changes are properly logged.

## **Database Management System**

### **BIM SEM 4**

#### **Transaction Processing**

Queries and other DML actions are grouped into transactions, which are units that must be executed atomically and in isolation from one another. Any query or modification action can be a transaction by itself. In addition, the execution of transactions must be durable, meaning that the effect of any completed transaction must be preserved even if the system fails in some way right after completion of the transaction. We divide the transaction processor into two major parts:

- I. A concurrency-control manager, or scheduler, responsible for assuring atomicity and isolation of transactions, and**
- II. A logging and recovery manager, responsible for the durability of transactions.**

#### **Task of Transaction Processing**

It is normal to group one or more database operations into a transaction, which is a unit of work that must be executed atomically and in apparent isolation from other transactions. In addition, a DBMS offers the guarantee of durability: that the work of a completed transaction will never be lost. The transaction manager therefore accepts transaction commands from an application, which tell the transaction manager when transactions begin and end, as well as information about the expectations of the application (some may not wish to require atomicity, for example). The transaction processor performs the following tasks:

- 1. Logging:** In order to assure durability, every change in the database is logged separately on disk. The log manager follows one of several policies designed to assure that no matter when a system failure or “crash” occurs, a recovery manager will be able to examine the log of changes and restore the database to some consistent state. The log manager initially writes the log in buffers and negotiates with the buffer manager to make sure that buffers are written to disk (where data can survive a crash) at appropriate times.
- 2. Concurrency control:** Transactions must appear to execute in isolation. But in most systems, there will in truth be many transactions executing at once. Thus, the scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed in their entirety, one-at-a-time. A typical scheduler does its work by maintaining locks on certain pieces of the database. These locks prevent two transactions from accessing the same piece of data in ways that interact badly. Locks are generally stored in a main-memory lock table, as suggested by Fig. above. The scheduler affects the execution of queries and other database operations by forbidding the execution engine from accessing locked parts of the database.
- 3. Deadlock resolution:** As transactions compete for resources through the locks that the scheduler grants, they can get into a situation where none can proceed because each need something another transaction has. The transaction manager has the responsibility to intervene and cancel (“rollback” or “abort”) one or more transactions to let the others proceed.

## **Database Management System**

### **BIM SEM 4**

#### **Storage and Buffer Management**

The data of a database normally resides in secondary storage; in today's computer systems "secondary storage" generally means magnetic disk. However, to perform any useful operation on data, that data must be in main memory. It is the job of the storage manager to control the placement of data on disk and its movement between disk and main memory.

In a simple database system, the storage manager might be nothing more than the file system of the underlying operating system. However, for efficiency purposes, DBMS's normally control storage on the disk directly, at least under some circumstances. The **storage manager** keeps track of the location of files on the disk and obtains the block or blocks containing a file on request from the buffer manager. The **buffer manager** is responsible for partitioning the available main memory into buffers, which are page-sized regions into which disk blocks can be transferred. Thus, all DBMS components that need information from the disk will interact with the buffers and the buffer manager, either directly or through the execution engine. The kinds of information that various components may need include:

1. **Data:** the contents of the database itself.
2. **Metadata:** the database schema that describes the structure of, and constraints on, the database.
3. **Log Records:** information about recent changes to the database; these support durability of the database.
4. **Statistics:** information gathered and stored by the DBMS about data properties such as the sizes of, and values in, various relations or other components of the database.
5. **Indexes:** data structures that support efficient access to the data.

=====End unit 1=====

#### **Unit 1: Introduction – Database Management Systems (LH 4)**

Purpose of Database Systems. Data Abstraction. Data Models: The E-R Model, The Object-Oriented Model, The Relational Model, The Network Model, The Hierarchical Model, Physical Data Models. Instances and Schemes. Data Independence. Database Administrator. Database Users. Application Architecture (One tier, two tier and n-tier). Overall Database System Structure and Components.