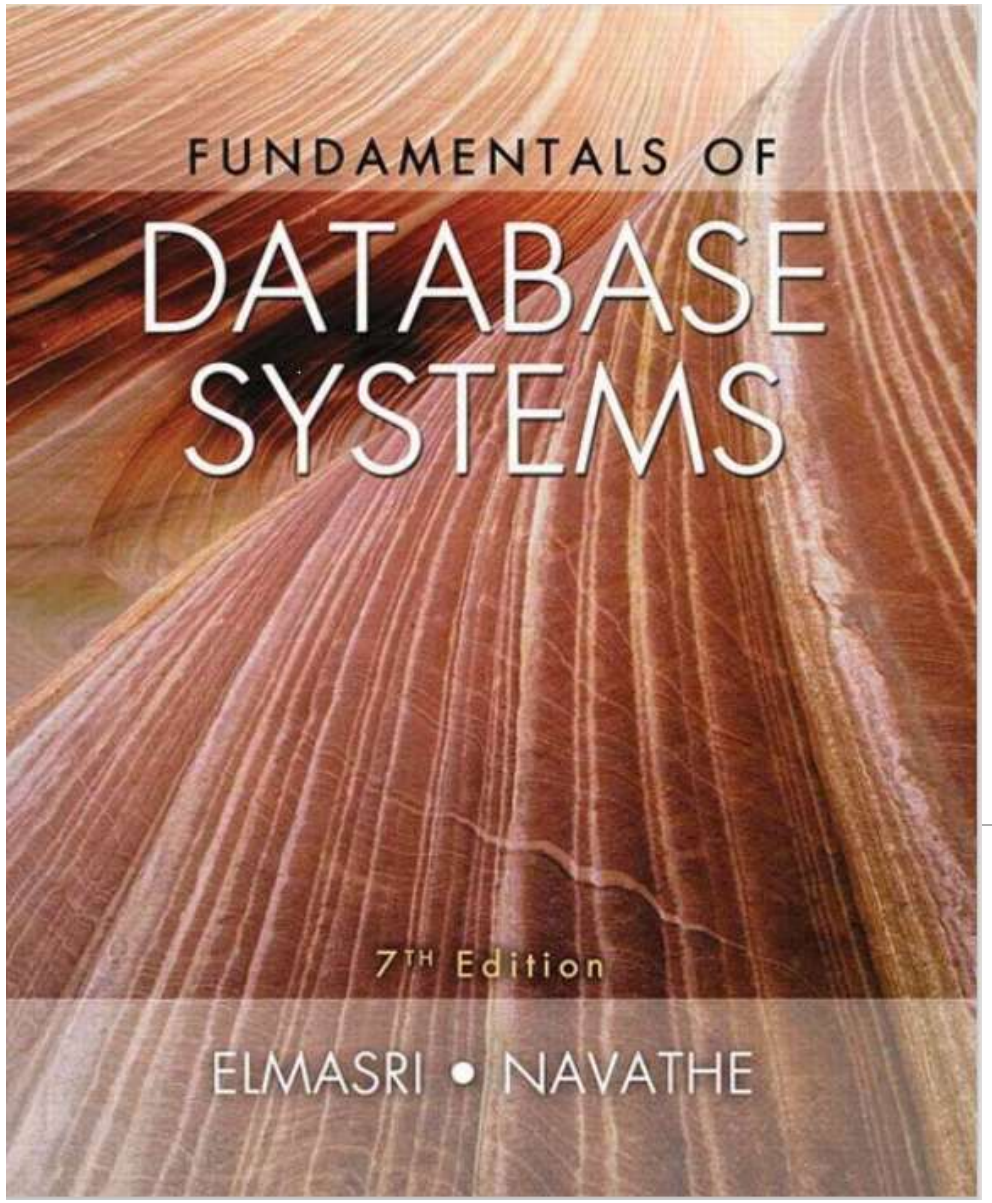


Database Management Systems

MODULE – 1

INTRODUCTION TO DBMS

Introduction

- **Data means information with an implicit meaning. E.g. Name of a person, Address, Phone number etc.**
- **Database is a place where related piece of information(Data) is stored and various operations can be performed on it.**
- **Basically, Database means collection of interrelated data where data can be easily accessed, managed and updated.**
- **Data is organized into rows, columns and tables to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added.**

Introduction

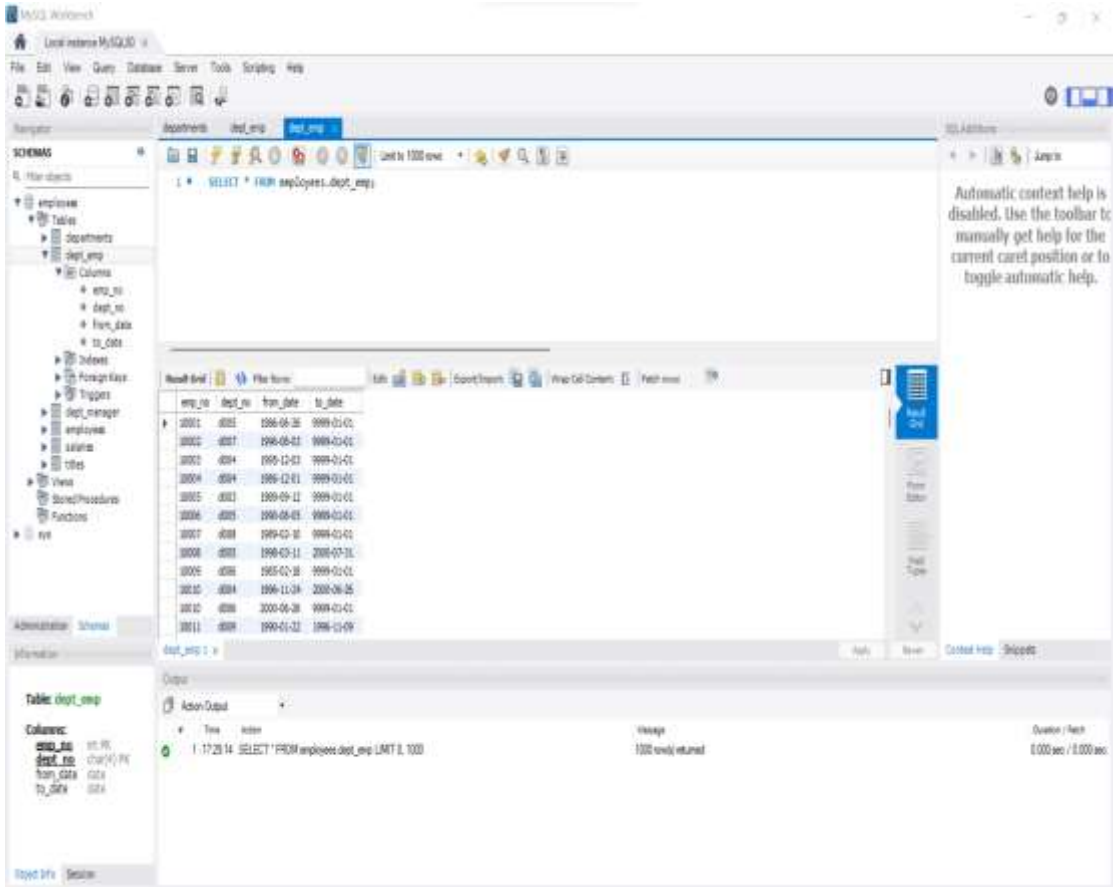
Introduction

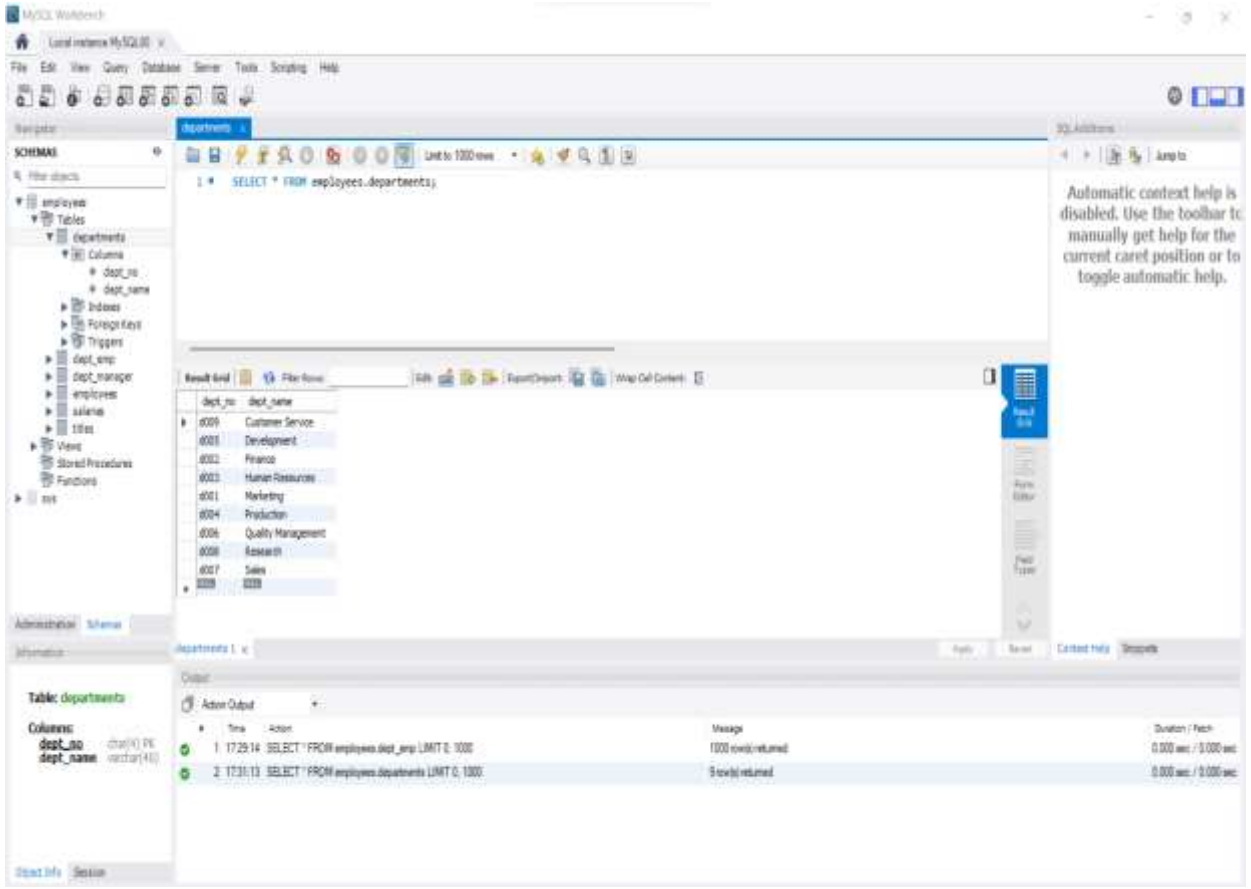
Traditional Databases applications includes numeric and textual data(structured data)

E.g.: Banking, Airline Ticket Reservation etc.

Nowadays we come across different types of data like posts, tweets, video clips , audio and images (Unstructured Data). To handle this kind of data there are more efficient databases such as NoSQL, Big Data System etc.

Introduction





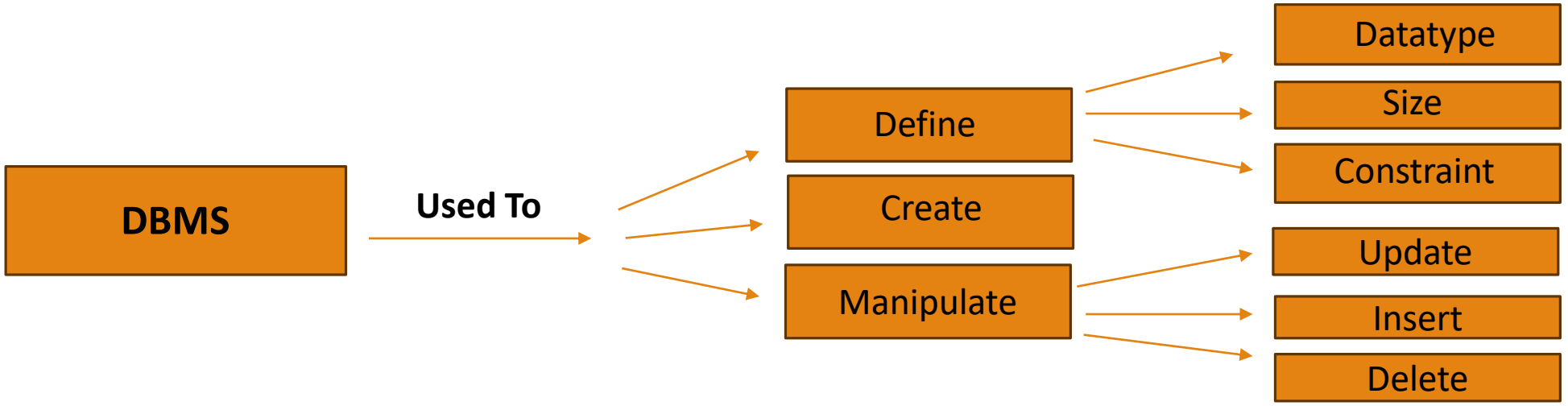
Introduction

- **A database management system (DBMS) is a computerized system that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.**

- **Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.**

- **Manipulating a database includes functions such as querying the database to retrieve specific data, updating the data etc.**

Introduction



Characteristics of Database Approach

- **The main characteristics of the database approach versus the file-processing approach are the following:**

1) **Self-describing nature of a database system**
2) **Insulation between programs and data, and data abstraction**
3) **Support of multiple views of the data**
4) **Sharing of data and multiuser transaction processing**

Characteristics of Database Approach

1) <u>Self-describing nature of a database system</u>

- **A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.**

- **This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called meta-data.**

- **The catalog is used by the DBMS software and also by database users who need information about the database structure.**

Characteristics of Database Approach

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Characteristics of Database Approach

2) Insulation between programs and data, and data abstraction

- **In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. We call this property program-data independence.**

Characteristics of Database Approach

In a file system if some changes are made in the file structure, then to handle these changes, more changes have to be made in all the programs that access this file.

For example you want to add a piece of data, say date of birth of students. Just adding it is not enough, the whole program will have to be re-written to make it work.

But In a Database system all you need to do is define another data item in the catalogue called date of birth and all the changes will be reflected and there is no need to change the whole program.

A DBMS provides users with a conceptual representation of data that does not include many of the details of how and where the data is stored which is called as data abstraction.

Characteristics of Database Approach

3) <u>Support of multiple views of the data</u>

- **A database typically has many types of users, each of whom may require a different perspective or view of the database.**
- **A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.**
- **A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.**
- **For example, one user of the database may be interested only in accessing and printing the transcript of each student. A second user, who is interested only in checking the total number of students who have joined in the current academic year.**

Characteristics of Database Approach

4) Sharing of data and multiuser transaction processing

- **A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database.**

- **The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.**

- **For example, Online bookings (ticketing, reservation systems, etc.). These types of applications are generally called on-line transaction processing (OLTP) applications.**

Actors on the Scene

- **For a small personal database, such as the list of addresses, one person typically defines, constructs, and manipulates the database, and there is no sharing.**
- **However, in large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds or thousands of users.**
- **In this section we identify the people whose jobs involve the day-to-day use of a large database; we call them the actors on the scene.**

1. **Database Administrators**
2. **Database Designers**
3. **End Users**
4. **System Analysts and Application Programmers (Software Engineers)**

Actors on the Scene

1. Database Administrators:

- **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).**
- **Database administrator (DBA): Responsible and Administering the database resources. The DBA is responsible for authorizing access to the database, for coordinating and monitoring its use.**
- **The DBA is accountable for problems such as breach of security or poor system response time.**

Actors on the Scene

2. Database Designers:

- **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 database designers to communicate with all prospective database users, in order to understand their requirements, and to come up with a design that meets these requirements.**
- **Database designers typically interact with each potential group of users and develop a view of the database that meets the data and processing requirements of the group.**
- **The final database design must be capable of supporting the requirements of all user groups.**

Actors on the Scene

3. End Users:

- **End users are the people who access to the database for querying, updating, and generating reports.**

Categories :

- **Casual end users**
- **Naive or parametric end users**
- **Sophisticated end users**
- **Stand-alone users**

Actors on the Scene

- **Casual end users - occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests. They are typically middle or high-level managers.**
- **Naive or parametric end users - Their main job is constantly querying the database, using standard types of queries and updates that have been carefully programmed and tested is called canned transactions.**

A few examples are:

Bank tellers check account balances and post withdrawals and deposits, Reservation clerks for airlines, hotels, and car rental companies check availability for a given request and make reservations

Actors on the Scene

- **Sophisticated end users** - **Include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS and to implement their applications to meet their complex requirements.**

- **Stand-alone users** - **Mostly maintain personal databases using ready-to-use packaged applications. An example is the user of a tax program that creates its own internal database. Another example is a user that maintains a database of personal photos and videos.**

Actors on the Scene

5. System Analysts and Application Developers - This category currently accounts for a very large proportion of the IT work force.

- **System Analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.**

- **Application Programmers: Implement the specifications developed by analysts and test and debug them before deployment.**

Workers Behind The Scene

- **In addition to those who use and administer a database, others are associated with the design, development, and operation of the DBMS software and system environment.**
- **These persons are typically not interested in the database content itself. We call them the workers behind the scene and they include the following categories:**

1) DBMS system designers and implementers

2) Tool developers

3) Operators and maintenance personnel

Workers Behind the Scene

- **DBMS system designers and implementers design and implement the DBMS modules as a software package.**
- **A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the query language processing and handling data recovery and security etc.**
- **The DBMS must interact with other system software, such as the operating system etc.**

Workers Behind the Scene

- **Tool developers design and implement tools - The software packages that facilitate database modeling and design, database system design, and improved performance.**
- **Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring etc. In many cases, independent software vendors develop and market these tools.**
- **Operators and maintenance personnel are responsible for the actual running and maintenance of the hardware and software environment for the database system.**

Advantages of Using the DBMS Approach

1) **Controlling Redundancy**
2) **Restricting Unauthorized Access**
3) **Providing Persistent Storage for Program Objects**
4) **Providing Storage Structures and Search Techniques for Efficient Query Processing**
5) **Providing Backup and Recovery**
6) **Providing Multiple User Interfaces**
7) **Representing Complex Relationships among Data**
8) **Enforcing Integrity Constraints**
9) **Permitting Inferencing and Actions Using Rules and Triggers**

Advantages of Using the DBMS Approach

1) Controlling Redundancy

- **In File Processing System, duplicate data is created in many places because all the programs have their own files. This creates data redundancy which in turns wastes labor and space.**
- **In Database Management System, all the files are integrated in a single database. The whole data is stored only once at a single place so there is no chance of duplicate data.**
- **For example: A student record in admission or examination section can contain duplicate values, but when they are converted into a single database, all the duplicate values are removed**

Advantages of Using the DBMS Approach

2) Restricting Unauthorized Access

- **Data security means protecting your precious data from unauthorized access. Data in database should be kept secure and safe to unauthorized modifications.**
- **Only authorized users should have the grant to access the database. There is a username set for all the users who access the database with password so that no other guy can access these information.**
- **DBMS always keep database tamperproof, secure and theft free.**

Advantages of Using the DBMS Approach

3) Providing Persistent Storage for Program Objects

- **Programming languages typically have complex data structures, such as structs or class definitions in C++ or Java.**
- **The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in permanent files or in the database.**
- **database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.**
- **Hence, The values of program variables in C++ can be stored permanently . Such program variables is said to be persistent, since it survives the termination of program execution**

Advantages of Using the DBMS Approach

4) Providing Storage Structures and Search Techniques for Efficient Query Processing

- **Database systems must provide capabilities for efficiently executing queries and updates. Because the database is typically stored on disk, the DBMS must provide specialized search techniques to speed up disk search for the desired records.**
- **In order to process the database records needed by a particular query, those records must be copied from disk to main memory. Therefore, the DBMS often has a buffering or caching module that maintains parts of the database that are frequently used in main memory buffers.**

Advantages of Using the DBMS Approach

5) Providing Backup and Recovery

- **A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.**
- **For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.**

Advantages of Using the DBMS Approach

6) Providing Multiple User Interfaces

Many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.

These include mobile apps for casual users; programming language interfaces for application programmers etc.

Advantages of Using the DBMS Approach

7) Representing Complex Relationships among Data

- **A database may include numerous varieties of data that are interrelated in many ways.**
- **A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.**

Advantages of Using the DBMS Approach

8) Enforcing Integrity Constraints

- **Most database applications have certain integrity constraints that must hold for the data.**
- **A DBMS should provide capabilities for defining and enforcing the constraints.**
- **The simplest type of integrity constraint involves specifying a data type for each data item.**

For example, defining the value of the Class data item within each STUDENT record must be a one-digit integer and that the value of Name must be a string of no more than 30 alphabetic characters.

- **Referential integrity constraint A referential integrity constraint is a rule that ensures that the values of a foreign key in one table match the values of a primary key in another table**
- **Another type of constraint specifies uniqueness on data item values, such as every course record must have a unique value for Course_number. This is known as uniqueness constraint.**

Advantages of Using the DBMS Approach

9) Permitting Inferencing and Actions Using Rules and Triggers

- **Some database systems provide capabilities for making conclusions about its data based on a set of well defined rules and facts. Such systems are called deductive database systems.**

- **In today's relational database systems, it is possible to associate triggers with tables. A trigger is a form of a rule activated by updates to the table, which results in performing some additional operations to some other tables, sending messages, and so on.**

- **More involved procedures to enforce rules are popularly called stored procedures.**

Advantages of Using the DBMS Approach

Triggers - A trigger in a database automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table.

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

A Brief History of Database Applications

- **Early Database Applications Using Hierarchical and Network Systems**

Many early database applications maintained records in large organizations such as corporations, universities, hospitals, and banks. In many of these applications, there were large numbers of records of similar structure.

For example, in a university application, similar information would be kept for each student, each course, each grade record, and so on.

The main types of early systems were based on main paradigm: hierarchical systems, network models etc.

These systems did not provide sufficient data abstraction and program-data independence capabilities.

A Brief History of Database Applications

Hierarchical Model Example :

Parent Table →

`employee` table

EmpNo	First Name	Last Name	Dept. Num
100	Almukhtar	Khan	10-L
101	Gaurav	Soni	10-L
102	Siddhartha	Soni	20-B
103	Siddhant	Soni	20-B

`computer` table

Serial Num	Type	User EmpNo
3009734-4	Computer	100
3-23-283742	Monitor	100
2-22-723423	Monitor	100
232342	Printer	100

← Child Table

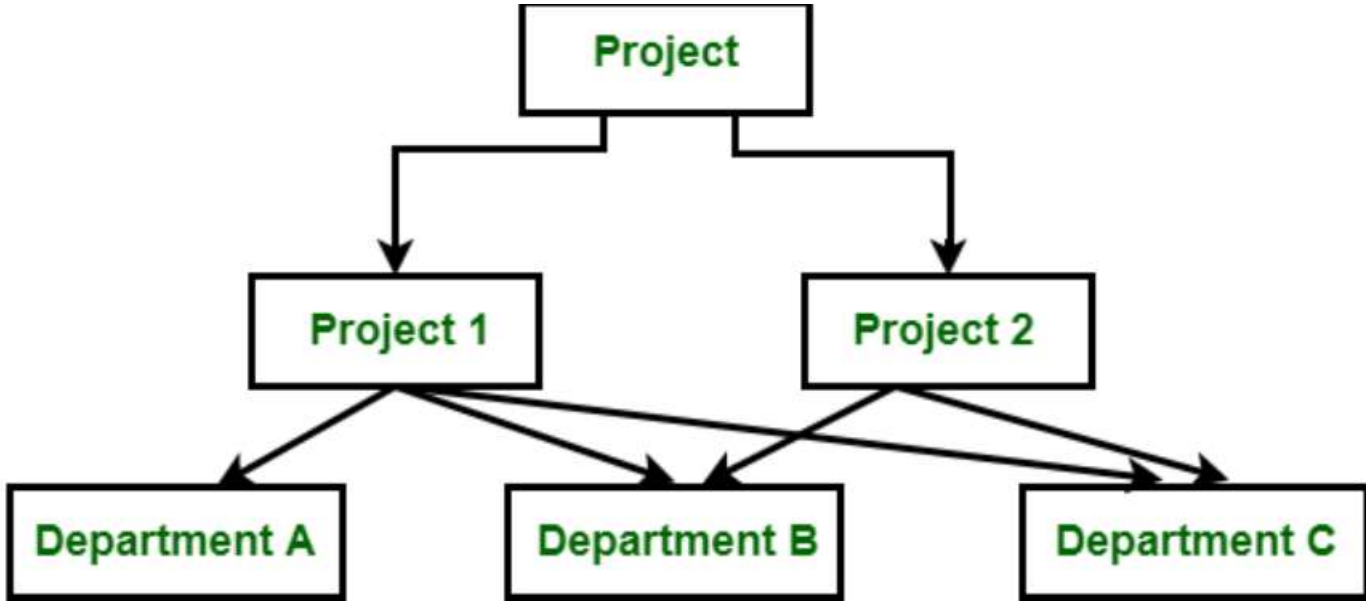
In this model, the employee data table represents the "parent" part of the hierarchy, while the computer table represents the "child" part of the hierarchy.

In this model, the child part point to the parent part. As shown, each employee may possess several pieces of computer equipment, but each individual piece of computer equipment may have only one employee owner. This represents one to many relationships.

A Brief History of Database Applications

Network Data Model:

It is the advance version of the hierarchical data model. In this child can have more than one parent. It implements 1:1, 1:n and also many to many relations.



A Brief History of Database Applications

- **Providing Data Abstraction and Application Flexibility with Relational Databases**

A relational database is a type of database that stores and allows access to data. These types of databases are referred to as "relational" because the data items within them have pre-determined relationships with one another. Data in a relational database is stored in tables.

Relational systems provided flexibility to develop new queries quickly and to reorganize the database as requirements changed.

Hence, data abstraction and program-data independence were much improved when compared to earlier systems.

Eventually, relational databases became the dominant type of database system for traditional database applications.

Relational databases now exist on almost all types of computers, from small personal computers to large servers

A Brief History of Database Applications

- **Object-Oriented Applications and the Need for More Complex Databases**

The emergence of object-oriented programming languages in the 1980s and the need to store and share complex, structured objects led to the development of object-oriented databases (OODBs).

Initially, OODBs were considered a competitor to relational databases, However the complexity of the model contributed to their limited use. They are now mainly used in specialized applications.

A Brief History of Database Applications

- **Interchanging Data on the Web for E-Commerce Using XML**

Users can create Web pages using a Web publishing language, such as Hyper Text Markup Language (HTML).

Much of the critical information on e-commerce Web pages is dynamically extracted data from DBMS whenever required.

A variety of techniques were developed to allow the interchange of dynamically extracted data on the Web for display on Web pages.

The extended Markup Language (XML) is one standard for interchanging data among various types of databases and Web pages.

A Brief History of Database Applications

- **Extending Database Capabilities for New Applications**

The following are some examples of these applications:

Scientific applications that store large amounts of data resulting from scientific experiments in areas such as the mapping of the human genome, and the discovery of protein structures.

Storage and retrieval of images, including scanned news or personal photographs, satellite photographic images, and images from medical procedures such as x-rays and MRI (magnetic resonance imaging) tests.

Storage and retrieval of videos, such as movies, and video clips from news or personal digital cameras.

Time series applications that store information such as economic data at regular points in time, such as daily sales and monthly gross national product figure

A Brief History of Database Applications

- **Emergence of Big Data Storage Systems and NOSQL Databases**

In the first decade of the twenty-first century, the proliferation of applications and platforms such as social media Web sites, large e-commerce companies led to a surge in the amount of data stored on large databases and massive servers.

New types of database systems were necessary to manage these huge databases.

The term NOSQL is generally interpreted as Not Only SQL, the systems that manage large amounts of data.

Data Models, Schemas, and Instances

A data model—a collection of concepts that can be used to describe the structure of a database which provides means to achieve data abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data.

1) High Level / conceptual data models

2) Representational data models

3) Low level / physical data models

Data Models, Schemas, and Instances

1) High Level / conceptual data models

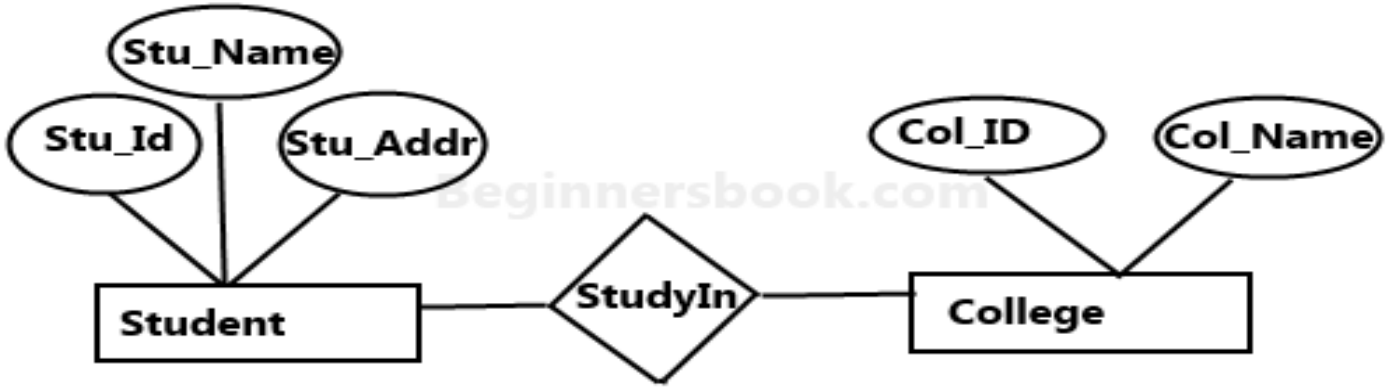
Conceptual data models use concepts such as entities, attributes, and relationships. An entity represents a real-world object or concept, such as an employee or a project from that is described in the database.

An attribute represents some property of interest that further describes an entity, such as the employee's name or salary.

A relationship among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.

Example : E – R Model.

Data Models, Schemas, and Instances



Sample E-R Diagram

Data Models, Schemas, and Instances

2) Representational Data Model

Between these two extremes(high level and low level) is a class of representational (or implementation) data models, which provide concepts that may be easily understood by end users.

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database.

The representational data model allows us to focus primarily, on the design part of the database. It is a theoretical concept whose practical implementation is done in Physical Data Model.

- **Relational Database Model**
- **Network Model**
- **Hierarchical Model**

Data Models, Schemas, and Instances

3) Low level / physical data models

low-level or physical data models provide concepts that describe the details of how data is stored. Concepts provided by physical data models are generally meant for computer specialists, not for end users.

It mainly describes the records stored, the types of records stored , The ordering of the records, access paths to those records etc.

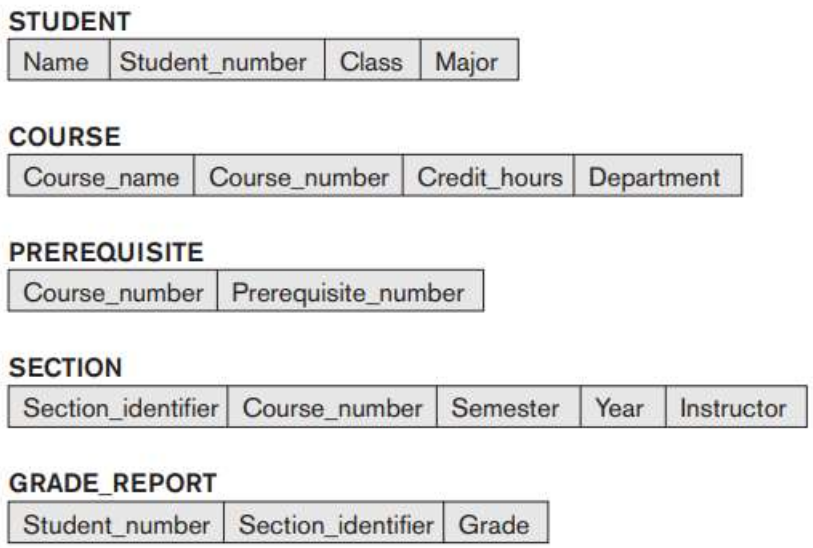
The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage devices.

Data Models, Schemas, and Instances

- **Schemas, Instances, and Database State**

In a data model, it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema, which is specified during database design and is not expected to change frequently.

Figure 2.1
Schema diagram for the database in Figure 1.2.



Data Models, Schemas, and Instances

We call each object in the schema—such as STUDENT or COURSE—a schema construct. The actual data in a database may change quite frequently.

The data in the database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database.

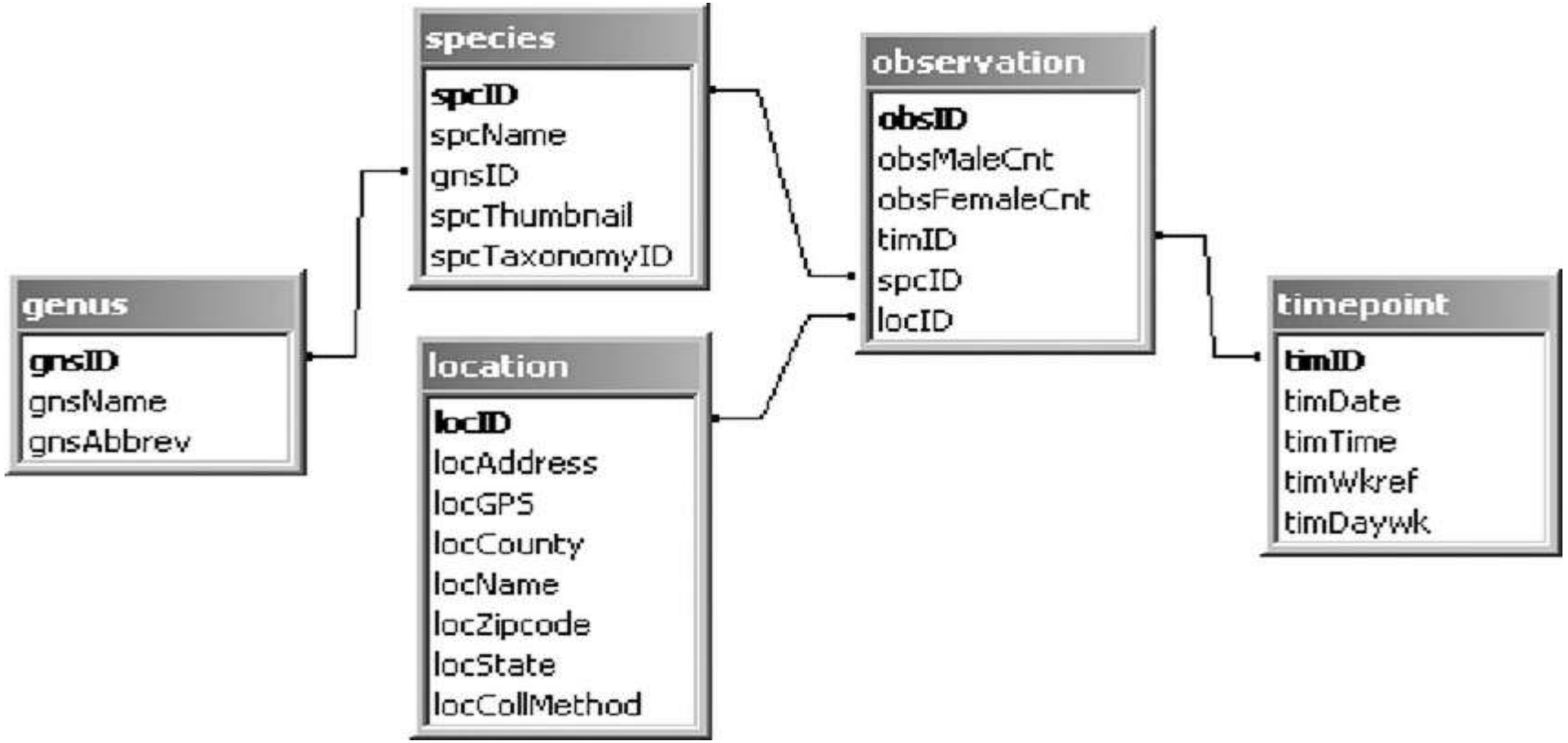
The distinction between database schema and database state is very important. When we define a new database, we specify its database schema only to the DBMS.

At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is first populated or loaded with the initial data.

The schema is not supposed to change frequently, it is not uncommon that changes occasionally need to be applied to the schema as the application requirements change.

For example, we may decide that another data item needs to be stored for each record in a file, such as adding the Date_of_birth to the STUDENT schema. This is known as schema evolution.

Data Models, Schemas, and Instances



Three-Schema Architecture and Data Independence

❖ **Three – Schema Architecture**

The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system.

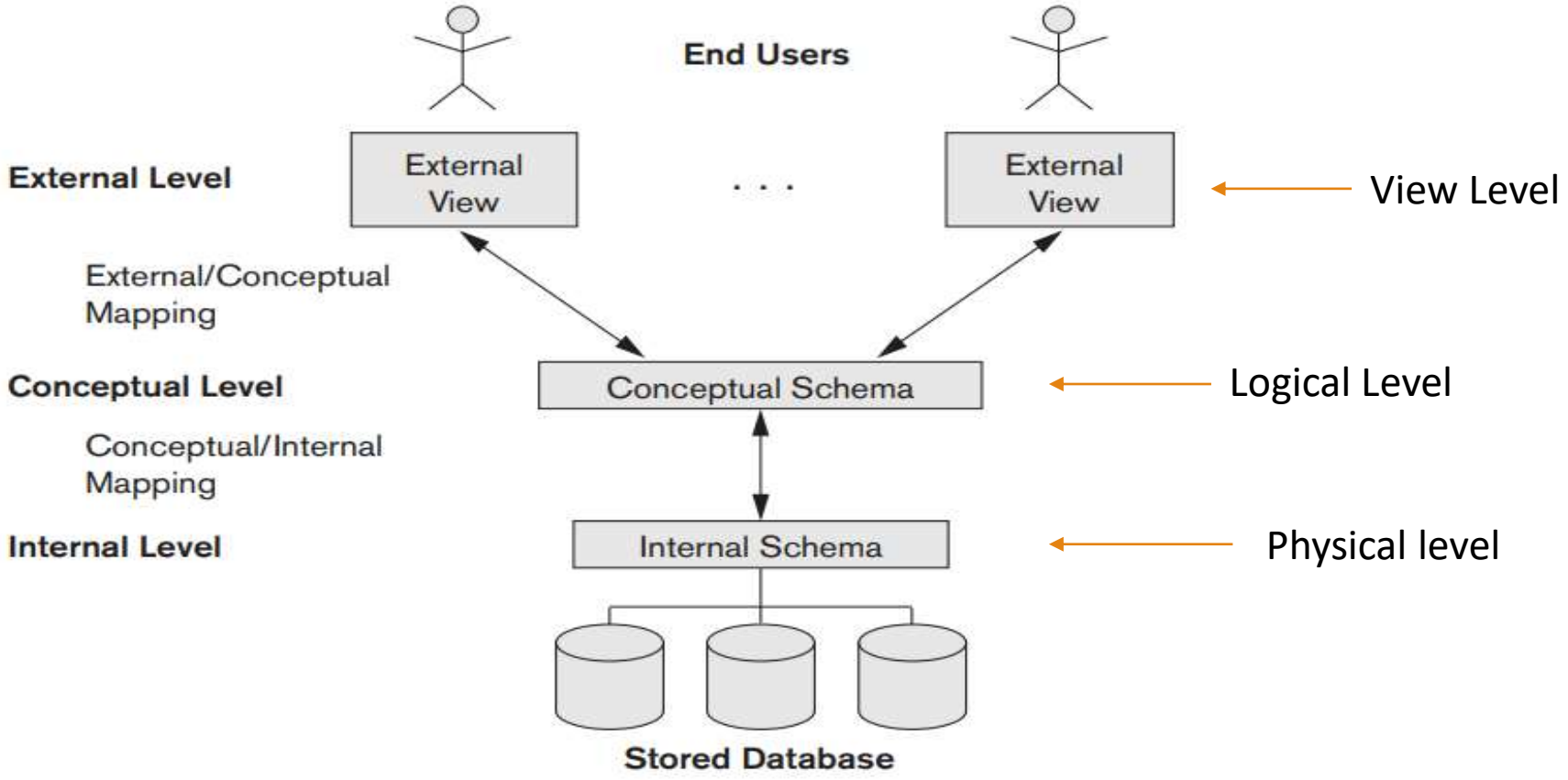
The goal of the three-schema architecture is to separate the user applications from the physical database.

1) **The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.**

2) **The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.**

3) **The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.**

Three-Schema Architecture and Data Independence



Three-Schema Architecture and Data Independence

Notice that the three schemas are only descriptions of data; the actual data is stored at the physical level only. In the three-schema architecture, each user group refers to its own external schema.

Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.

If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.

Three-Schema Architecture and Data Independence

The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

We can define two types of data independence:

1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).

2. Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.

Changes to the internal schema may be needed because some physical files were reorganized to improve the performance of retrieval or update.

Database Languages and Interfaces

A DBMS has appropriate languages and interfaces to express database queries and updates. Database languages can be used to read, store and update the data in the database.

- **Data Definition Language (DDL) - DDL stands for Data Definition Language. It is used to define database structure or pattern. It is used to create schema, tables etc. in the database. levels, the DDL is used to specify the conceptual schema only.**
- **Data Manipulation Language (DML) - DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.**
- **Data Control Language (DCL) - includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.**

Database Languages and Interfaces

❖ **Data Definition Language (DDL)**

- **CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).**
- **DROP: This command is used to delete the database and tables within the database.**
- **ALTER: This is used to alter the structure of the database.**
- **TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.**
- **RENAME: This is used to rename an table existing in the database.**

Database Languages and Interfaces

- **Data Manipulation Language (DML)**

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

- **Data Control Language (DCL)**

- **GRANT: This command gives users access privileges to the database.**
- **REVOKE: This command withdraws the user's access privileges given by using the GRANT command.**

Database Languages and Interfaces

- **An interface is a program that allows users to query the DBMS without writing the code in query language.**
- **An interface can be used to manipulate the database either for adding the data, or deleting some data, or updating some data, or even for viewing the data present in the database.**
- **There are different types of interfaces for different types of users:**

1) **Form Based Interface**
2) **Menu Based User Interface**
3) **Graphical User Interface**
4) **Natural Language Interface**
5) **Speech Input and output Interface**
6) **Interface for Parametric Users**
7) **Interface for the DBA**
8) **Apps for Mobile Devices**
9) **Keyword-based Database Search**

Database Languages and Interfaces

User-friendly interfaces supported by a DBMS may include the following:

1) Form Based Interface : A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.

2) Menu-based Interfaces for Web Clients or Browsing : These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.

Pull-down menus are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory manner.

Database Languages and Interfaces

3) Graphical User Interface - A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms.

4) Natural Language Interface - These interfaces accept requests written in English or some other language and attempt to understand them. A natural language interface usually has its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words.

The natural language interface refers to the words in its schema, as well as to the set of standard words in its dictionary, that are used to interpret the request.

If the interpretation is successful, a high-level query corresponding to the natural language request is created and submitted to the DBMS for processing.

Database Languages and Interfaces

5) Speech Input and output Interface - Applications such as inquiries for flight arrival/departure, and credit card account information, are allowing speech for input and output to enable customers to access this information.

The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.

6) Interface for Parametric Users - Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. For example, a bank teller is able to use interfaces for repetitive transactions such as account deposits or withdrawals, or balance inquiries.

Database Languages and Interfaces

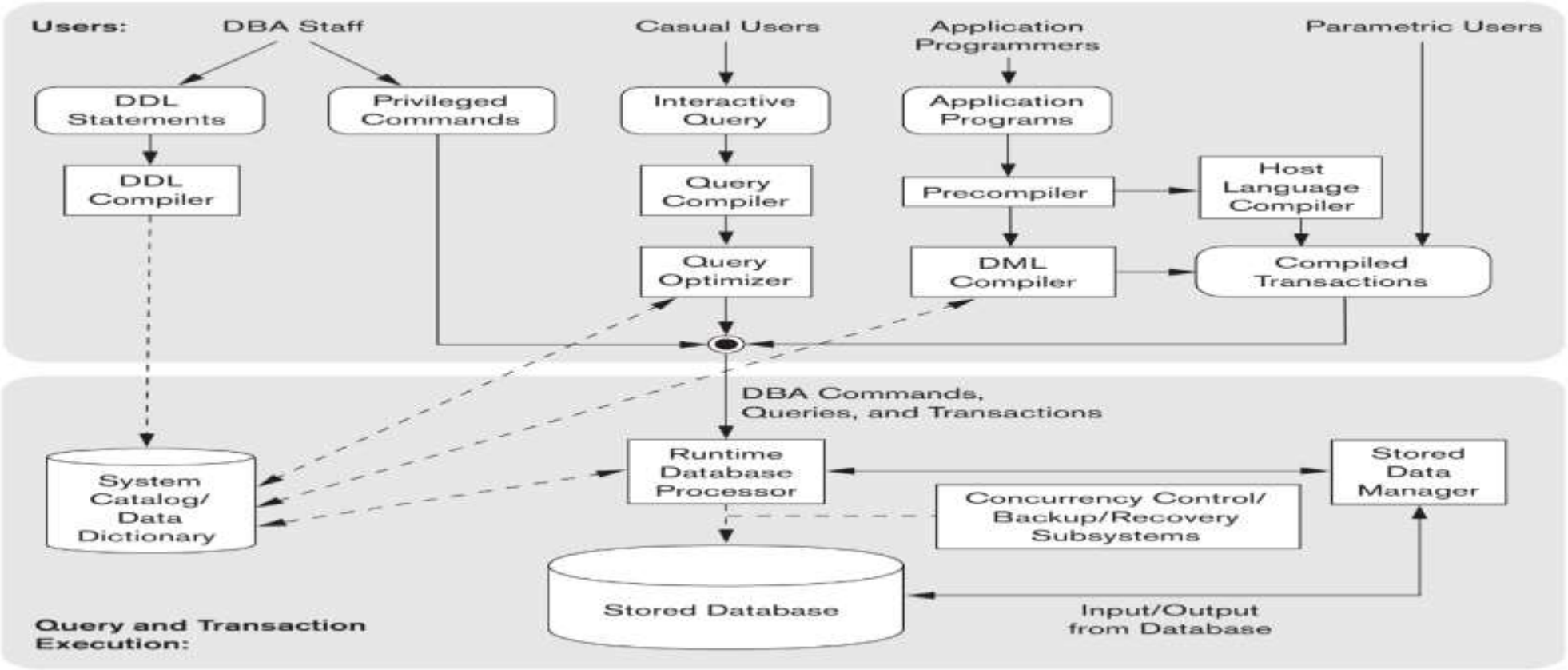
7) Interfaces for the DBA - Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for creating accounts, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

8) Apps for Mobile Devices - These interfaces present mobile users with access to their data. For example, banking, reservations, and insurance companies, among many others, provide apps that allow users to access their data through a mobile phone or mobile device.

9) Keyword-based Database Search - These are somewhat similar to Web search engines, which accept strings of words and match them with documents at specific sites or Web pages (for engines like Google).

The Database System Environment

DBMS Component Modules:



The Database System Environment

- **The database system component modules are the different components that are made available in order to take care of the transactions in the database.**
- **The figure describes the various types of users such as DBA Staff, casual users, application programmers and parametric users.**
- **DBA Staff are responsible for creating the database, tables and deleting the database etc. So, DBA Staff will be focusing on DDL statements and also on privileged commands. The DDL compiler will check the correctness of DDL Commands.**
- **For example ,If the database is created using DDL Statements, the schema for the database has been created and it is stored in the database catalog.**

The Database System Environment

- **In the case of casual users, If any request for the data is given by the casual users. They can use the interfaces like menu based interfaces, form based interfaces etc.**
- **The request given by the users are converted in to an interactive query which is compiled by the query complier.**
- **Query optimizer is used to optimize the queries to increase the performance by rearranging the operations. The optimizer will interact with the database catalog to optimize the query.**
- **Application programmers will develop the application programs using high level languages like java, c++ etc. Pre compiler is used to separate the database specific queries from the application program. These database specific queries is fed into DML Compiler.**

The Database System Environment

- **The remaining portion of the application program has been fed into host language compiler which checks the correctness of the remaining portion of the application program.**
- **The compiled transactions will be containing the combination of compiled versions of both host language compilers and also the DML Complier.**
- **All the queries given by the different users are fed into runtime database processor for further processing. The database processor will be having interaction with the database catalog and finally executes the queries.**

The Database System Environment

- **Stored data manager will maintain the input/output transactions between the disk and main memory. The data that is needed might be stored in the disk so to retrieve the data from the disk, there has to be a transaction between disk and main memory.**
- **For this dbms is having subsystem named stored data manager which controls the transactions between disk and main memory. It interacts with the runtime database**
- **There are other subsystems namely the concurrency control systems, backup systems which controls the concurrent access to the database by the users and provides database backup facilities respectively.**

The Database System Environment

Database System Utilities:

- **Loading**
- **Backup**
- **Database Storage Reorganization**
- **Performance monitoring**

The Database System Environment

loading – A loading utility is used to load existing data files—such as text files into the database. Usually, the current (source) format of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database.

Backup - A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database.

Database storage reorganization - This utility can be used to reorganize a set of database files into different file organizations and create new access paths to improve performance.

Performance monitoring – This utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files to improve performance.

Centralized and Client/Server Architectures for DBMS

- **Centralized DBMS Architecture**

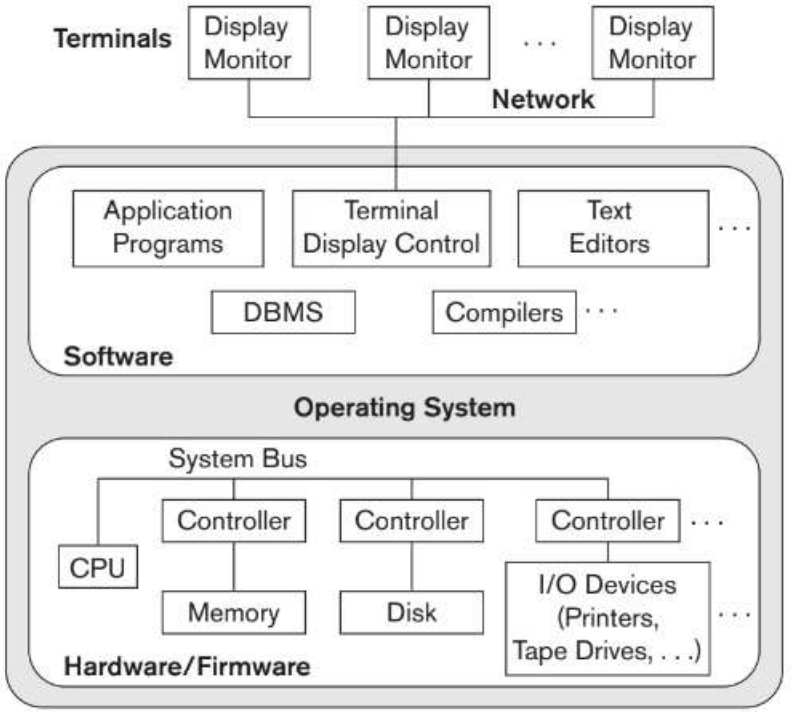


Figure 2.4
A physical centralized architecture.

Centralized and Client/Server Architectures for DBMS

- **In Centralized DBMS Architecture, systems, most users accessed the DBMS via computer terminals that did not have processing power and only provided display capabilities.**
- **Therefore, all processing was performed remotely on the computer system housing the DBMS, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.**
- **As prices of hardware declined, most users replaced their terminals with PCs and workstations, and more recently with mobile devices.**
- **Gradually, DBMS Systems started to exploit the available processing power at the user side, which led to the client/server DBMS architectures.**

Centralized and Client/Server Architectures for DBMS

There are two types in the client/server architectures:

1) Logical Two Tier Client/Server Architecture.

2) Physical Two Tier Client/Server Architecture.

- **Logical Two Tier Client/Server Architecture.**

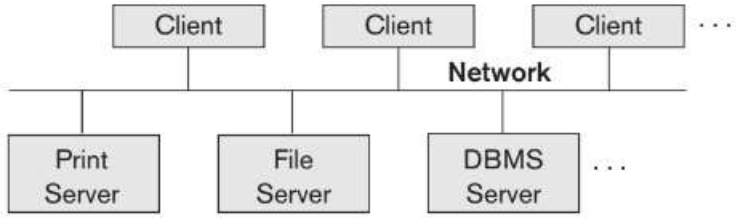


Figure 2.5
Logical two-tier client/server architecture.

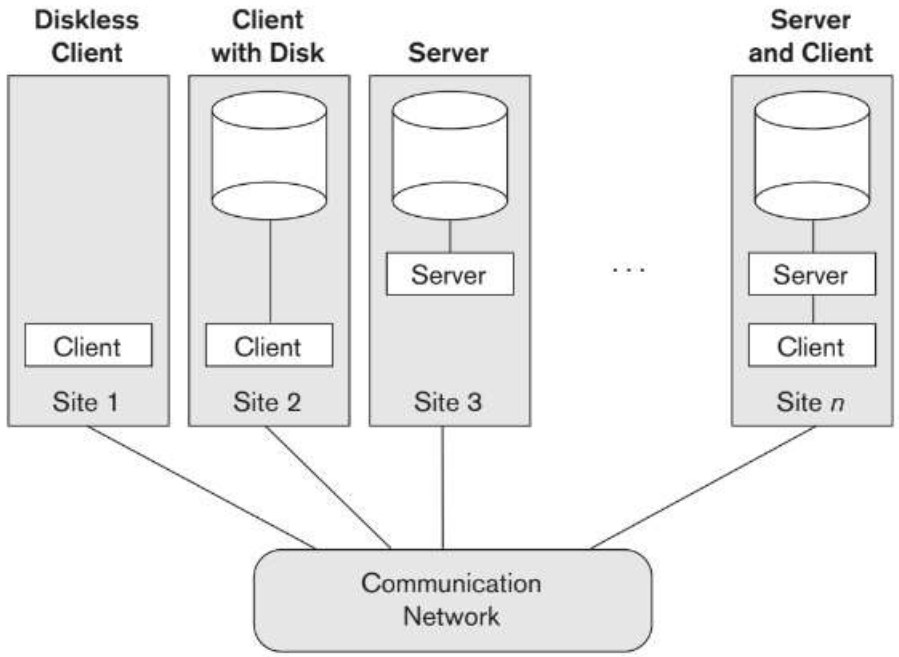
Centralized and Client/Server Architectures for DBMS

- **The two tier architecture is mentioned here because there are two machines. One is the client machine and the other is the server. where client machine is used by the user and the server is used to fulfil the requests given by the user. Both the server and client machine are connected through a network.**
- **The client/server architecture was developed to deal with computing environments in which a large number of file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network.**
- **The idea is to define specialized servers with specific functionalities for example: separate server for files, separate servers for DBMS etc.**
- **The resources provided by specialized servers can be accessed by many client machines.**
- **The client machine also provides the users with appropriate user interfaces. So that the users can interact with the servers.**

Centralized and Client/Server Architectures for DBMS

- **Physical Two Tier Client/Server Architecture**

Figure 2.6
Physical two-tier client/server architecture.



Centralized and Client/Server Architectures for DBMS

- **Physical Two Tier Client/Server Architecture**
  - **Some machines would be client sites only, other machines would be dedicated servers, and others would have both client and server functionality**
  - **A client machine provides user interface capabilities and local processing. A server is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.**
  - **In general, some machines install only client software, others only server software, and still others may include both client and server software.**
  - **The disadvantages of both the types of client/server architectures is that the client is given a direct access to the database which results in security issues.**

Centralized and Client/Server Architectures for DBMS

Three-Tier and n-Tier Architectures for Web Applications

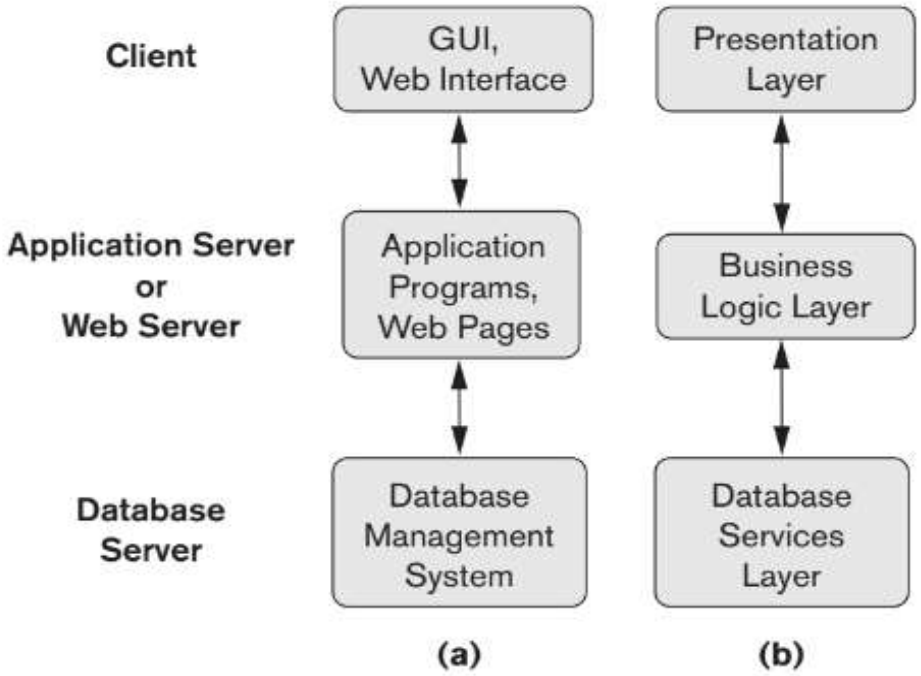


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Centralized and Client/Server Architectures for DBMS

- **To overcome the disadvantages of two tier architecture, The three tier architecture was introduced which did not give the direct access of database to the client.**
- **Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server**
- **This intermediate layer or middle tier is called the application server or the Web server, depending on the application**
- **This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server.**
- **It can also improve database security by checking a client's credentials before forwarding a request to the database server.**
- **The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing processed data from the database server to the clients**

Centralized and Client/Server Architectures for DBMS

- **Thus, the user interface, application rules, and data access act as the three tiers. The presentation layer displays information to the user and allows data entry.**
- **The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS**
- **The bottom layer includes all data management services. The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side**
- **If business logic layer is divided into multiple layer, then called as n-tier architecture**

THANKYOU