RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Database Systems

Course Code: CS52

Credits: 3:1:0

UNIT 1

Term: October 2021– February 2022

Reference:
Elmasri, R., Shamkant B. Navathe, R.
Fundamentals of Database Systems

RAMAIAH
Institute of Technology

Unit I – Chapter 2

- Introduction
- Characteristics of Database approach
- Actors on the Scene
- Workers behind the scene
- Advantages of using DBMS approach
- Data models, schemas and instances
- Three-schema architecture and data independence
- Database languages and interfaces
- The database system environment
- Centralized and client-server architectures
- Classification of Database Management systems,
- Entity-Relationship Model:
- Using high level Conceptual data models for Database Design
- A Sample Database Application
- Entity types, Entity sets Attributes and Keys
- Relationship types, Relationship Sets,
- Roles and Structural Constraints
- Weak Entity Types.

RAMAIAH Institute of Technology

DATA MODELS, SCHEMAS, AND INSTANCES (1)

- The architecture of DBMS
- **Monolithic systems** - the whole DBMS software package is **one tightly integrated system**
- Modern DBMS packages that are **modular in design**, with a **client/server system architecture**.
- **Client module** - runs on user workstation or personal computer. Application programs and user interfaces that access the database run in the client module. User-friendly interfaces such as **forms- or menu based GUls** (Graphical User Interfaces).
- **Server module** - handles **data storage, access, search**, and other functions. Mainframe computers are used.

RAMAIAH Institute of Technology

DATA MODELS, SCHEMAS, AND INSTANCES (2)

- A **data model-** is a collection of concepts that can be used to describe the structure (data types, relationships, and constraints) of a database.
- Data models include a set of basic operations for specifying **retrievals and updates** on the database.
- **User-defined** operation - COMPUTE_GPA
- **Basic data model operations** - insert, delete, modify, or retrieve

RAMAIAH Institute of Technology

DATA MODELS, SCHEMAS, AND INSTANCES (3)
Categories of Data Models (1)

- **High-level or conceptual data models** - close to the way many users perceive data, for **end users.**
- **low-level or physical data models** - describe the details of **how data is stored in the computer** - meant for computer specialists.
- **Representational or implementation data models** - **hide some details of data storage** but can be implemented on a computer system in a direct way.
- **Conceptual data models** - use concepts such as **entities, attributes, and relationships**.
- **Entity** represents **a real-world object or concept**, such as an employee or a project, that is described in the database.
- An a**ttribute** 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 two or more entities

RAMAIAH Institute of Technology

DATA MODELS, SCHEMAS, AND INSTANCES (4)
Categories of Data Models (2)

- **Physical data models** describe how **data is stored as files in the computer** by representing information such as **record formats, record orderings, and access paths.**
- An **access path** is a structure that **makes the search for particular database records efficient**.
- Representational or implementation data models include **relational data model**, the **network** and **hierarchical** models. They represent data by using record structures and hence are sometimes called record-based data models.
- **Object data models** are also frequently utilized as **high-level conceptual models**, particularly in the software engineering domain.

RAMAIAH Institute of Technology

DATA MODELS, SCHEMAS, AND INSTANCES (5)
Schemas, Instances, and Database State(1)

- **Database schema** - The **description of a database** which is specified during database design and is not expected to change frequently.
- **Schema diagram** - A **displayed schema** is called a schema diagram.
- **Schema construct** - each **object** in the schema ex:- STUDENT or COURSE.
- **Database state or snapshot** - The data in the **database at a particular moment in time**. It is also called the *current* set of occurrences or instances in the database
- The schema is sometimes called the **intension**, and a database state an **extension** of the schema.
- **Schema evolution** – **changes** made to schema, for ex:- adding new attribute DoB.

DATA MODELS, SCHEMAS, AND INSTANCES (6)
Schemas, Instances, and Database State(2)

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 2.1
Schema diagram for the database in Figure 1.2.

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (1)

The Three Schema Architecture (1)

- The goal of the three-schema architecture, is to **separate the user applications and the physical database.**
- In this architecture, schemas can be defined at the following **three levels**:

1. The internal level

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

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (2)
The Three Schema Architecture (2)

- **The conceptual level**
  - has a **conceptual schema**, which **describes the structure of the whole database** for a community of users.
  - **hides the details of physical storage** structures and concentrates on **describing entities, data types, relationships, user operations, and constraints.**
  - **representational data model** is used to describe the conceptual schema
  - This *implementation conceptual schema* is often based on a *conceptual schema design* in a high-level data model.

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (3)
The Three Schema Architecture (3)

3. The external or view level

- includes a number of **external schemas or user views**.
- 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.
- implemented using a **representational data model**, possibly based on an **external schema design** in a highlevel data model.

Most DBMSs **do not separate** the three levels completely.

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (4)
The Three Schema Architecture (4)

- In a DBMS based on the three-schema architecture, each user group refers only 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 (5)

The Three Schema Architecture (5)

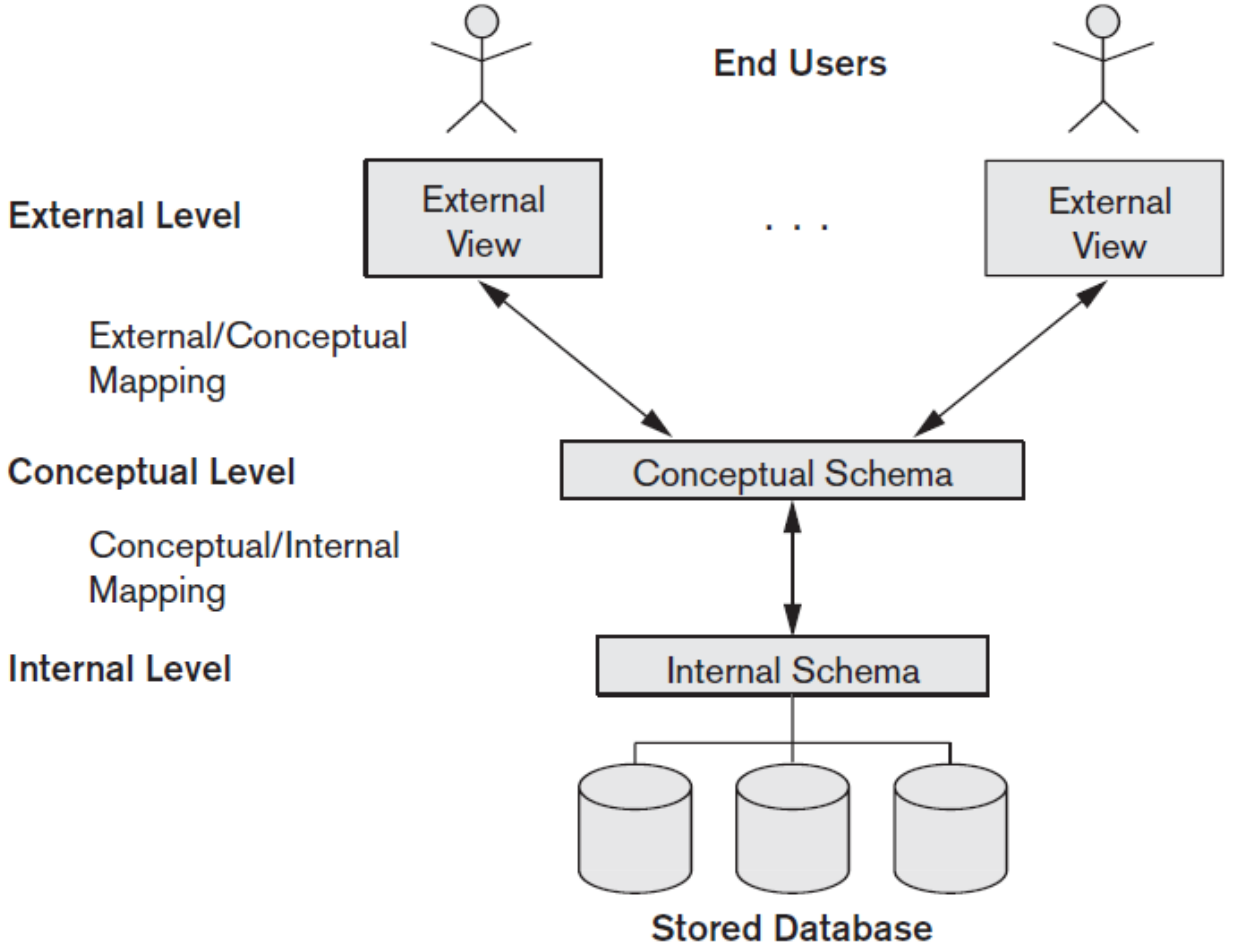

Figure 2.2
The three-schema architecture.

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (6)
Data Independence (1)

- Data independence 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**.

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).
- Only the **view definition and the mappings** need be **changed** in a DBMS that supports logical data independence

THREE-SCHEMA ARCHITECTURE AND DATA INDEPENDENCE (7)
Data Independence (2)

2. Physical data independence

- is the **capacity to change the internal schema without having to change the conceptual schema**.
- Changes to the internal schema may be needed because some **physical files had to be reorganized-** for example, by creating additional access structures-to improve the performance of retrieval or update.
- If the same data as before remains in the database, we should not have to change the conceptual schema. ex:- change of access structure.

- Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; **only the *mapping* between the two levels is changed**.

DATABASE LANGUAGES AND INTERFACES (1)
DBMS LANGUAGES (1)

- **Data definition language (DDL),** is used by the DBA and by database designers to define both conceptual and internal schemas when there is no strict separation of the levels.
- The DBMS will have a DDL compiler which process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.
- **Storage Definition Language (SDL),** is used to specify the internal schema.
- The mappings between the two schemas may be specified in either DDL or SDL languages.
- **View Definition Language (VDL),** is used to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.

DATABASE LANGUAGES AND INTERFACES (2)
DBMS LANGUAGES (2)

- **Data Manipulation Language (DML)** is used to manipulated database which includes retrieval, insertion, deletion, and modification of the data.
- A **comprehensive integrated language** is used that includes constructs for conceptual schema definition, view definition, ami data manipulation.
- **SQL relational database language** is a comprehensive database language which represents a combination of DDL, VDL, and DML.
- There are two main types of DMLs.
- A **high-level or nonprocedural DML** can be used on its own to specify complex database operations in a concise manner.
- Many DBMSs allow high-level DML statements either to be entered **interactively** from a display monitor or terminal or to be **embedded** in a general-purpose programming language.

DATABASE LANGUAGES AND INTERFACES (3)
DBMS LANGUAGES (3)

- In the latter case, DML statements must be identified **within the program** so that they can be extracted by a **precompiler** and processed by the DBMS.
- A **low-level or procedural DML must be embedded** in a general-purpose programming language. This type of DML typically retrieves **individual records** or objects from the database and processes each separately.
- Hence, it needs to use programming language constructs, such as **looping**, to retrieve and process each record from a set of records.
- Low-level DMLs are also called **record-at-a-time DMLs** because of this property.
- High-level DMLs, such as **SQL**, can specify and retrieve many records in a single DML statement and are hence called **set-at-a-time or set-oriented DMLs**.

DATABASE LANGUAGES AND INTERFACES (4)
DBMS LANGUAGES (4)

- A query in a high-level DML often specifies **which data to retrieve rather than how to retrieve it**; hence, such languages are also called **declarative**.
- Whenever DML commands, whether high level or low level, are **embedded** in a general-purpose programming language, that language is called the **host language** and the DML is called the data **sublanguage.**
- On the other hand, a **high-level DML** used in a **stand-alone interactive** manner is called a query language.
- **Casual end users** typically use a **high-level query language** to specify their requests, whereas **programmers** use the **DML in its embedded form.**

DATABASE LANGUAGES AND INTERFACES (5)
DBMS Interface (1)

- **Menu-Based Interfaces for Web Clients or Browsing** – present the user with lists of options, called **menus**, that lead the user through the formulation of a request. **No need to memorize the specific commands** and syntax of a query language; rather, the query is composed step by step by picking options from a menu that is displayed by the system. **Pull-down menus** are a very popular technique in **Web-based user interfaces**.

- **Forms-Based Interfaces** - displays a **form** to each user. **Users can fill out** all of the form entries to **insert new data**, or they fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries. Forms are usually designed and programmed for **naive users** as interfaces to canned transactions. Many DBMSs have **forms specification languages**, which are special languages that help programmers specify such forms.

DATABASE LANGUAGES AND INTERFACES (6)
DBMS Interface (2)

- **Graphical User Interfaces -** displays a schema to the user in **diagrammatic form**. The user can then specify a query by manipulating the diagram. In many cases, GUIs utilize both **menus and forms**. Most GUIs use a **pointing device**, such as a **mouse**, to pick certain parts of the displayed schema diagram.

- **Natural Language 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, to **interpret the request**. If the interpretation is successful, the interface generates a **high-level query** corresponding to the natural language request and submits it to the DBMS for processing; otherwise, **a dialogue is started with the user to clarify the request.**

DATABASE LANGUAGES AND INTERFACES (7)
DBMS Interface (3)

- **Interfaces for Parametric Users -** a small set of abbreviated commands is included, with the goal of **minimizing the number of keystrokes** required for each request. For example, **function keys** in a terminal can be programmed to initiate the various commands. This allows the parametric user to proceed with a minimal number of keystrokes.

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

THE DATABASE ENVIRONMENT (1)
DBMS Component Modules (1)

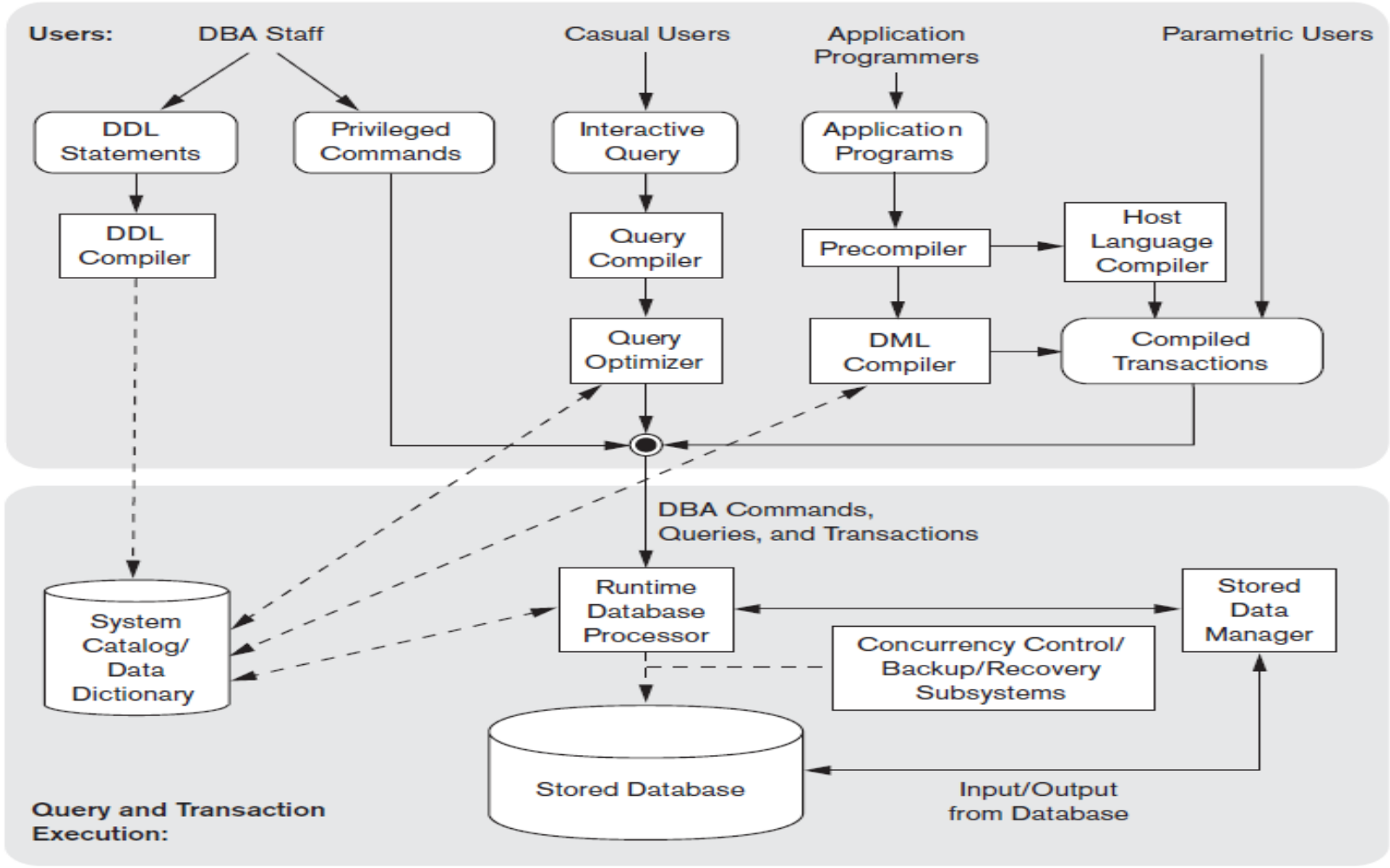

Figure 2.3
Component modules of a DBMS and their interactions.

THE DATABASE ENVIRONMENT (2)
DBMS Component Modules (2)

- Figure 2.3 illustrates, in a simplified form, the typical **DBMS components.**
- The **database** and the DBMS **catalog** are usually **stored on disk**.
- A higher-level **stored data manager module** of the DBMS controls **access to DBMS information that is stored on disk**, whether it is part of the database or the catalog.
- The stored data manager may **use basic os services for carrying out lowlevel data transfer** between the disk and computer main storage, but it **controls** other aspects of data transfer, such as **handling buffers in main memory**.
- The **DDL compiler processes schema definitions**, specified n the DDL, and **stores** descriptions of the schemas (**meta-data**) in the DBMS **catalog**. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints, in addition to many other types of information that are needed by the DBMS modules.

THE DATABASE ENVIRONMENT (3)
DBMS Component Modules (3)

- The **runtime database processor** handles **database accesses at runtime**; it receives retrieval or update operations and carries them out on the database.
- The **query compiler** handles **high-level queries** that are entered **interactively**. It **parses, analyzes, and compiles or interprets a query** by creating database access code, and then **generates calls to the runtime processor** for executing the code.
- The **precompiler extracts DML commands from an application program** written in a host programming language. These commands are sent to the **DML compiler** for compilation into object code for database access. The **rest** of the program is sent to the **host language compiler**. The **object codes** for the **DML commands** and the **rest of the program are linked**, forming a canned transaction whose executable code includes calls to the runtime database processor.

THE DATABASE ENVIRONMENT (4)
DBMS Component Modules (4)

- The **DBMS interacts with the operating system** when **disk accesses**-to the **database** or to the **catalog**-are needed.
- If the computer system is **shared by many users**, the **os will schedule** DBMS disk access requests and DBMS processing along with other processes.
- On the other hand, if the computer system is mainly **dedicated to running the database server**, the **DBMS** will **control main memory buffering of disk pages**.
- The **DBMS also interfaces** with **compilers** for general-purpose host programming languages, and with **application servers** and **client programs** running on separate machines through the **system network interface**.
-

THE DATABASE ENVIRONMENT (5)
Database System Utilities (1)

- DBMSs have database utilities that help the **DBA** in managing the database system. Common utilities have the following types of functions:

- ***Loading:*** A loading utility is used to **load existing data files-such as text files or sequential files-into the database**. Usually, the current **(source) format** of the data and the **desired (target) database** file structure are **specified** to the **utility**, which then **automatically reformats** the data and stores it in the database. Some **vendors** are offering products that generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas). Such tools are also called **conversion tools**.

- ***Backup:*** A backup utility creates a **backup copy of the database**, usually by dumping the entire database onto **tape**. The backup copy can be used to restore the database in case of **catastrophic failure**. **Incremental backups** are also often used, where only changes since the previous backup are recorded. Incremental backup is **more complex but saves space**.

THE DATABASE ENVIRONMENT (6)
Database System Utilities (2)

- ***File reorganization:*** This utility can be used **to reorganize a database file** into a different file organization to **improve performance**.
- ***Performance monitoring:*** **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.

Other utilities may be available for **sorting files, handling data compression, monitoring access by users, interfacing with the network**, and performing other functions.

THE DATABASE ENVIRONMENT (7)
Tools, Application Environments, and Communications Facilities (1)

- Other tools are often available to database designers, users, and DBAs.
- CASE tools are used in the design phase of database systems.
- Expanded **data dictionary (or data repository**) system. In addition to storing catalog information about schemas and constraints, the data dictionary stores other information, such as **design decisions, usage standards, application program descriptions, and user information.** Such a system is also called an **information repository.** This information can be accessed *directly* by users or the DBA when needed.
- A data dictionary utility is similar to the DBMS catalog, but it includes a **wider variety of information** and is **accessed mainly by users** rather than by the DBMS software.

THE DATABASE ENVIRONMENT (8)
Tools, Application Environments, and Communications Facilities (2)

- **Application development environments**, such as the **PowerBuilder (Sybase) or JBuilder (Borland) system**, are becoming quite popular. These systems provide an environment for **developing database applications** and include facilities that help in many facets of database systems, including **database design, GUI development, querying and updating, and application program development.**

- The DBMS also needs to interface with **communications software**, whose function is to **allow users at locations remote from the database system site to access the database** through computer terminals, workstations, or their local personal computers. These are **connected** to the database site through data communications hardware such as **phone lines**, **long-haul networks, local area networks, or satellite communication devices.**

- Many commercial database systems have **communication packages** that work with the DBMS. **The integrated DBMS and data communications system is called a DB/DC system**. In addition, some **distributed DBMSs** are **physically distributed over multiple machines**. **LANs** are used to connect them.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(1)
Centralized DBMSs Architecture(1)

- Earlier architectures for DBMSs used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality.
- The reason was that most users accessed such systems via computer terminals that did not have processing power and only provided display capabilities.
- As prices of hardware declined, most users replaced their terminals with PCs and workstations.
- At first, DBMS itself was still a **centralized** DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on one machine.
- Figure 2.4 illustrates the physical components in a centralized architecture. Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(2)
Centralized DBMSs Architecture(2)

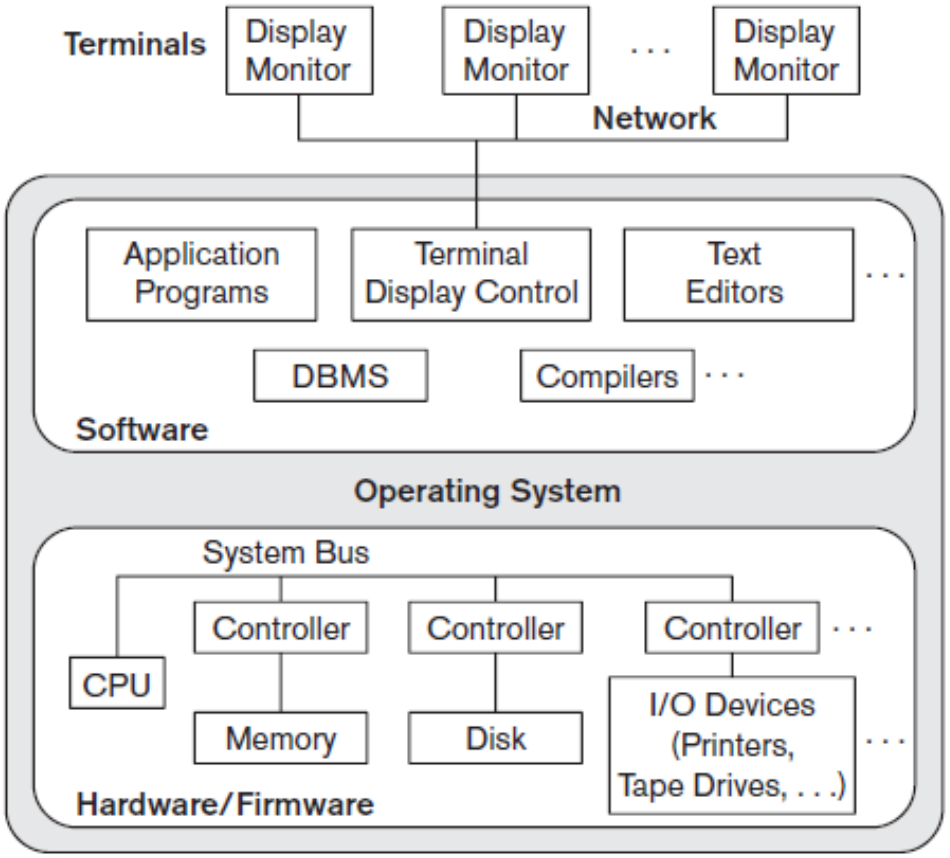

Figure 2.4
A physical centralized architecture.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(3)
Basic Client/Server Architectures (1)

- The **client/server architecture** was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, data base 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, it is possible to connect a number of PCs or small workstations as clients to a **file server** that maintains the files of the client machines.
- Another machine can be designated as a **printer server** by being connected to various printers; all print requests by the clients are forwarded to this machine.
- **Web servers** or **e-mail servers** also fall into the specialized server category.
- The resources provided by specialized servers can be accessed by many client machines. The **client machines** provide the user with the **appropriate interfaces to utilize these servers**, as well as with local processing power to run local applications.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(4)
Basic Client/Server Architectures (2)

- This concept can be carried over to other software packages, with specialized programs—such as a **CAD (computer-aided design) package**—being stored on specific server machines and being made accessible to multiple clients.

- Figure 2.5 illustrates client/server architecture at the logical level

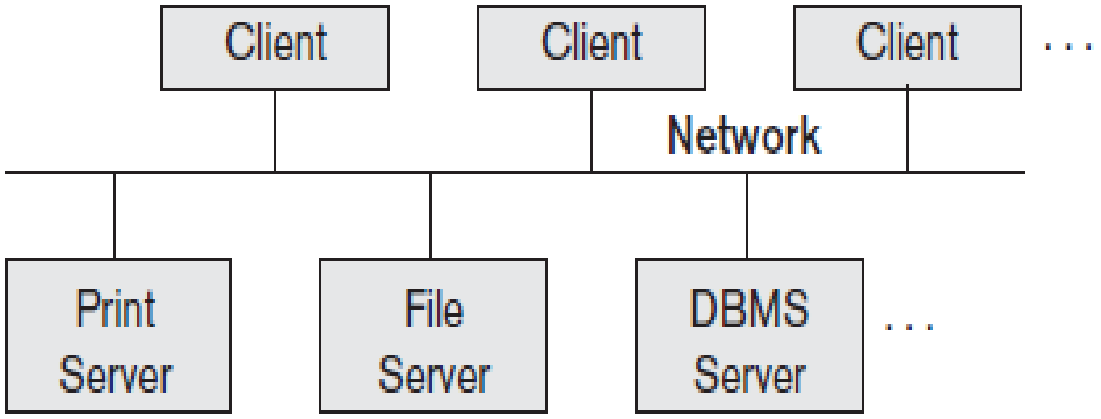

Figure 2.5
Logical two-tier client/server architecture.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(5)
Basic Client/Server Architectures (3)

- Figure 2.6 is a simplified diagram that shows the physical architecture. Some machines would be client sites only (for example, diskless workstations or workstations/PCs with disks that have only client software installed). Other machines would be dedicated servers, and others would have both client and server functionality.
- The concept of client/server architecture assumes an underlying framework that consists of many PCs and workstations as well as a smaller number of mainframe machines, connected via LANs and other types of computer networks.
- A **client** in this framework is typically a user machine that provides user interface capabilities and local processing. When a client requires access to additional functionality— such as database access—that does not exist at that machine, it connects to a server that provides the needed functionality.
- 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.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(6)
Basic Client/Server Architectures (4)

- In general, some machines install only client software, others only server software, and still others may include both client and server software, as illustrated in Figure 2.6.
- However, it is more common that client and server software usually run on separate machines.
- Two main types of basic DBMS architectures were created on this underlying client/server framework: **two-tier** and **three-tier**.

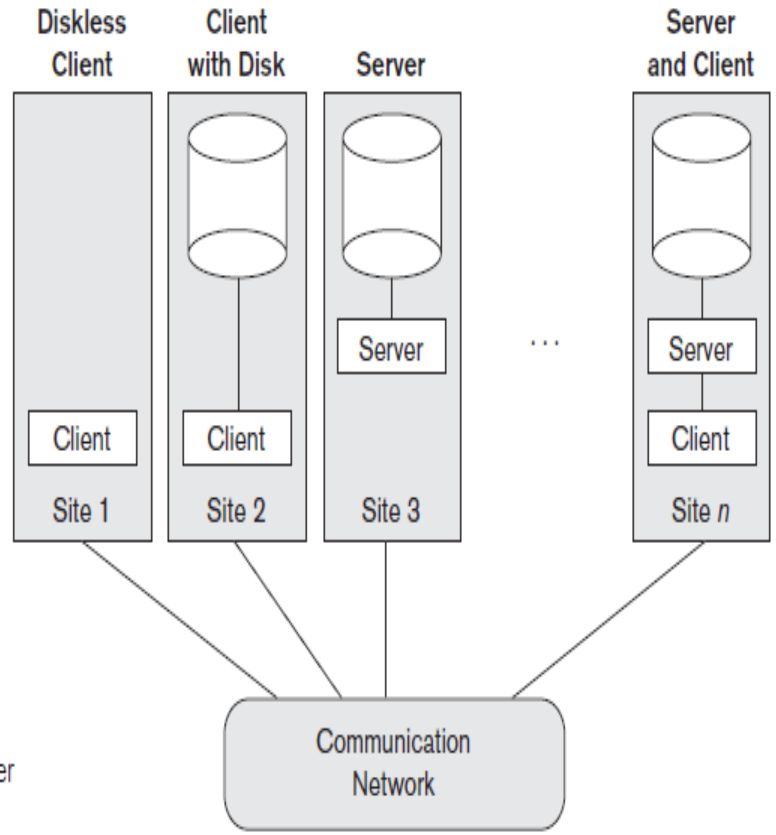

Figure 2.6
Physical two-tier client/server architecture.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(7)
Two-Tier Client/Server Architectures for DBMSs (1)

- The architecture is called **two-tier architectures** because the software components are distributed over two systems: **client and server**.
- The **advantages** of this architecture are its **simplicity and seamless compatibility** with existing
- systems.
- **SQL** provides a standard language for relational database management systems (**RDBMSs**), the **query and transaction functionality** related to SQL processing remained on the **server side** and is called a **query server** or **transaction server.**
- In an RDBMS, the server is also often called an **SQL server**.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(8)
Two-Tier Client/Server Architectures for DBMSs (2)

- The **user interface programs and application programs** can run on the **client side.**
- When DBMS access is required, the **program establishes a connection to the DBMS** on the server side, once the connection is created, the client program can communicate with the DBMS.
- A standard called **Open Database Connectivity** (**ODBC**) provides an **application programming interface** (**API**), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed.
- Most DBMS vendors provide **ODBC drivers** for their systems.
- A client program can actually **connect to several RDBMSs** and send query and transaction requests using the **ODBC API**, which are then processed at the server sites.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(9)
Two-Tier Client/Server Architectures for DBMSs (3)

- Any **query results are sent back to the client program**, which can process and display the results as needed.
- A related standard for the Java programming language, called **JDBC**, has also been defined. This allows Java client programs to access one or more DBMSs through a standard interface.
- Object-oriented DBMSs - software modules of the DBMS were divided between client and server in a more integrated way.
- For example, the **server level** may include the part of the DBMS software responsible for handling data storage on disk pages, local concurrency control and recovery, buffering and caching of disk pages, and other such functions.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(10)
Two-Tier Client/Server Architectures for DBMSs (4)

- Meanwhile, the **client level** may handle the user interface; data dictionary functions; DBMS interactions with programming language compilers; global query optimization, concurrency control, and recovery across multiple servers; structuring of complex objects from the data in the buffers; and other such functions.
- In this approach, the client/server interaction is more tightly coupled and is done internally by the DBMS modules which reside either on the client or the server side.
- In such a client/server architecture, the server has been called a **data server** because it provides data in disk pages to the client. This data can then be structured into objects for the client programs by the client-side DBMS software.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(11)
Three-Tier and n-Tier Architectures for Web Applications (1)

- Many Web applications use an architecture called the **three-tier architecture**, which adds an intermediate layer between the client and the database server, as illustrated in Figure 2.7(a).
- 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.

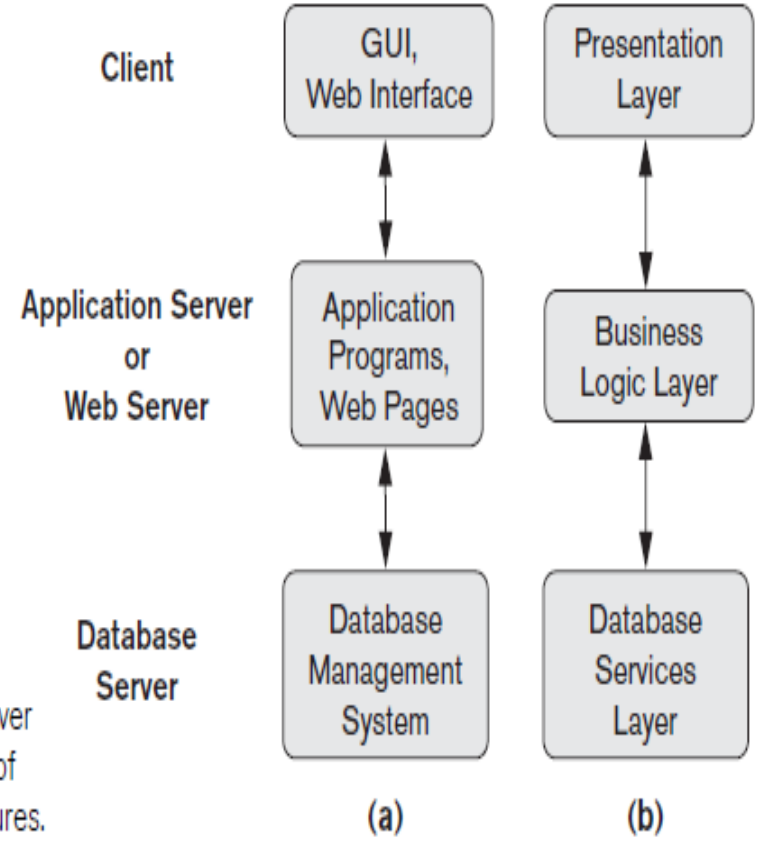

Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(12)
Three-Tier and n-Tier Architectures for Web Applications (2)

- It can also **improve database security** by checking a **client's credentials** before forwarding a request to the database server.
- **Clients** contain **GUI interfaces** and some additional application-specific business rules.
- 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 (partially) processed data from the database server to the clients, where it may be processed further and filtered to be presented to users in GUI format.
- Thus, the ***user interface, application rules,*** **and** ***data access*** act as the three tiers.
- Figure 2.7(b) shows another architecture used by database and other application package vendors.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(13)
Three-Tier and n-Tier Architectures for Web Applications (3)

- 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.
- ***n*-tier architectures**, where *n* may be four or five tiers.

CENTRALIZED AND CLIENT/SERVER ARCHITECTURES FOR DBMS(14)
Three-Tier and n-Tier Architectures for Web Applications (4)

- Vendors of **ERP (enterprise resource planning)** and **CRM (customer relationship management)** packages often use a ***middleware layer***, which accounts for the front-end modules (clients) communicating with a number of back-end databases (servers).
- Advances in **encryption and decryption** technology make it safer to transfer sensitive data from server to client in encrypted form, where it will be decrypted. The latter can be done by the hardware or by advanced software. This technology gives higher levels of data security, but the network security issues remain a major concern.
- Various technologies for **data compression** also help to transfer large amounts of data from servers to clients over wired and wireless networks.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (1)

- Several criteria are normally used to classify DBMSs viz., data model, number of users, number of sites over which data is distributed, cost, type of access path, general purpose or special purpose
- The first is the **data model** on which the DBMS is based.
- We can categorize DBMSs based on the data model: **relational, object, object-relational, hierarchical, network**, and other.
- The main data model used in many current commercial DBMSs is the **relational data model**.
- The basic **relational data model** represents a database as a collection of tables, where each table can be stored as a separate file.
- Most relational databases use the high-level query language called SQL and support a limited form of user views.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (2)

- The **object data model** defines a database in terms of objects, their properties, and their operations. Objects with the same structure and behavior belong to a **class**, and classes are organized into **hierarchies** (or **acyclic graphs**). The operations of each class are specified in terms of predefined procedures called **methods**.

- Relational DBMSs have been extending their models to incorporate object database concepts and other capabilities; these systems are referred to as **object-relational** or **extended relational systems**.

- The **XML model** has emerged as a standard for **exchanging data over the Web**, and has been used as a basis for implementing several prototype native XML systems.

- XML uses **hierarchical tree structures**. It combines database concepts with concepts from **document representation models**. Data is represented as elements; with the **use of tags**, data can be nested to create complex hierarchical structures.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (3)

- Two older, historically important data models, now known as **legacy data models**, are the network and hierarchical models.

- The **network model** represents data as record types and also represents a limited type of 1:N relationship, called a **set type**.

- A 1:N, or one-to-many, relationship relates one instance of a record to many record instances using some pointer linking mechanism in these models.

- Figure 2.8 shows a network schema diagram for the database of Figure 2.1, where record types are shown as rectangles and set types are shown as labelled directed arrows.

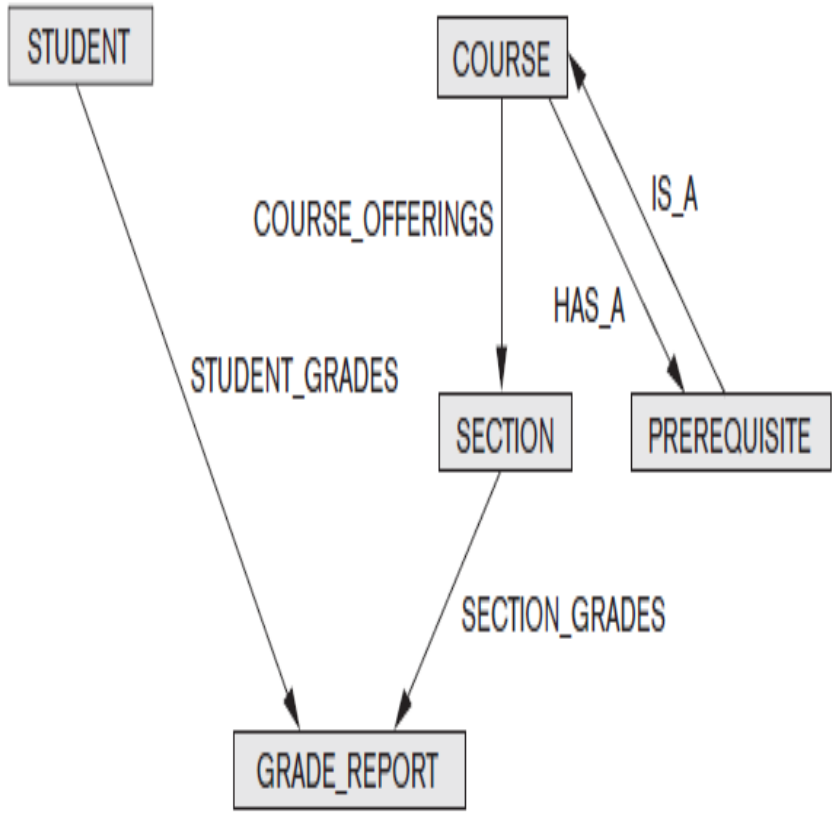

Figure 2.8
The schema of Figure 2.1 in network model notation.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (4)

- The network model, also known as the **CODASYL DBTG model,** has an associated record-at-a-time language that must be embedded in a host programming language.
- The network DML was proposed in the 1971 **Database Task Group (DBTG)** Report as an extension of the COBOL language.
- It is prominently used by **IDMS, IMAGE, and SUPRA DBMSs** today.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (5)

- The **hierarchical model** represents data as hierarchical tree structures.
- Each hierarchy represents a number of related records. There is no standard language for the
- hierarchical model. A popular hierarchical DML is DL/1 of the IMS system.
- Examples of hierarchical DBMSs include **IMS (IBM)** and some other systems like System **2K (SAS Inc.)** and **TDMS.**
- IMS is still used at governmental and industrial installations, including hospitals and banks, although many of its users have converted to relational systems.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (6)

- The second criterion used to classify DBMSs is the **number of users** supported by the system.
- **Single-user systems** support only one user at a time and are mostly used with PCs. **Multiuser systems**, which include the majority of DBMSs, support concurrent multiple users.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (7)

- The third criterion is the **number of sites** over which the database is distributed.
- A DBMS is **centralized** if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site.
- A **distributed** DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network.
- **Homogeneous** DDBMSs use the same DBMS software at all the sites, whereas **heterogeneous** DDBMSs can use different DBMS software at each site.
- It is also possible to develop **middleware software** to access several autonomous preexisting
- databases stored under heterogeneousDBMSs. This leads to a **federated** DBMS (or **multidatabase system**), in which the participating DBMSs are loosely coupled and have a degree of local autonomy.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (8)

- The fourth criterion is **cost**. **open source (free) DBMS** products like **MySQL and PostgreSQL** that are supported by third-party vendors with additional services.
- The main RDBMS products are available as free examination **30-day copy versions** as well as personal versions, which may **cost under $100** and allow a fair amount of functionality.
- The giant systems are being **sold in modular form** with components to handle distribution, replication, parallel processing, mobile capability.
- Furthermore, they are sold in the form of licenses—**site licenses** allow unlimited use of the database system with any number of copies running at the customer site.
- Another type of license **limits the number of concurrent users** or the number of user seats at a **location**.
- **Standalone single user versions** of some systems like **Microsoft Access** are **sold per copy** or included in the overall configuration of a desktop or laptop.
- In addition, **data warehousing and mining features**, as well as support for additional data types, are made available at **extra cost**.
- It is possible to pay **millions of dollars** for the installation and maintenance of **large database** systems **annually**.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (9)

- We can also classify a DBMS on the basis of the **types of access path** options for storing files.
- One well-known family of DBMSs is based on **inverted file structures.**

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS (10)

- Finally, a DBMS can be **general purpose** or **special purpose**.
- When performance is a primary consideration, a **special-purpose DBMS** can be designed and built for a specific application; such a system cannot be used for other applications without major changes. ex:- **airline reservations and telephone directory systems**.
- These fall into the category of **online transaction processing** (**OLTP**) systems, which must support a large number of concurrent transactions without imposing excessive delays.
-

Review Questions (1)

2.1. Define the following terms: *data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML, SQL,VDL, query language, host language, data sublanguage, database utility, catalog, client/server architecture.*

2.2. Discuss the main categories of data models.

2.3. What is the difference between a database schema and a database state?

2.4. Describe the three-schema architecture. Why do we need mappings between

schema levels? How do different schema definition languages support this architecture?

2.5. What is the difference between logical data independence and physical data independence?

Review Questions (2)

2.6. What is the difference between procedural and nonprocedural DMLs?

2.7. Discuss the different types of user-friendly interfaces and the types of users who typically use each.

2.8. With what other computer system software does a DBMS interact?

2.9. What is the difference between the two-tier and three-tier client/server architectures?

2.10. Discuss some types of database utilities and tools and their functions.

Exercises

2.11. Think of different users for the database of Figure 1.2. What types of applications would each user need? To which user category would each belong, and what type of interface would each need?

2.12. Choose a database application with which you are familiar. Design a schema and show a sample database for that application, using the notation of Figures 2.1 and 1.2. What types of additional information and constraints would you like to represent in the schema? Think of several users for your database, and design a view for each.

END OF CHAPTER 2