Contents

Unit - 1: Introduction

1. Introduction

Databases and database technology have had a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, social media, engineering, medicine, genetics, law, education, and library science. The word database is so commonly used that we must begin by defining what a database is.

2. Database

A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. Nowadays, this data is typically stored in mobile phones, which have their own simple database software. This data can also be recorded in an indexed address book or stored on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the mini-world or the universe of discourse (UoD). Changes to the mini world are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
- A database is designed, built, and populated with data for a specific purpose.

A database is a collection of data, typically describing the activities of one or more related organizations.

For example, a university database might contain information about the following:

- Entities such as students, faculty, courses, and classrooms.
- Relationships between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

3. Database Management Systems (DBMS)

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. (Ramez Elmasri, 2016)

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

(Abraham Silberschatz, 2020)

A database management system, or DBMS, is software designed to assist in maintaining and utilizing large collections of data, and the need for such systems, as well as their use, is growing rapidly. (Raghu Ramakrishnan)

Figure 1: Some Popular DBMS

4. Database System Applications

Database systems, first developed in the 1960s to manage commercial data, have evolved from handling simple, structured records to supporting complex, variable information for large enterprises. Their core function is to manage and protect valuable data, which is often more crucial than physical assets for modern organizations. Whether dealing with standardized university records or diverse social network content, all database applications share the need to efficiently manage large, multi-user collections of information. Modern systems balance efficiency for structured data and flexibility for less organized content, simplifying access through abstraction so users need not understand underlying complexities. This unified, abstracted management enables enterprises to integrate diverse data types and support their operations effectively.

Here are few representative applications of DBMS.

- **Enterprise Information**

    - **Sales**: For customer, product, and purchase information.
    - **Accounting**: For payments, receipts, account balances, assets, and other accounting information.
    - **Human resources**: For information about employees, salaries, payroll taxes, and benefits, and for the generation of paychecks.
- **Manufacturing**: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- **Banking and Finance**
    - **Banking**: For customer information, accounts, loans, and banking transactions.
    - **Credit card transactions**: For purchases on credit cards and the generation of monthly statements.
    - **Finance**: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also, for storing real-time market data to enable online trading by customers and automated trading by the firm.
- **Universities**: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- **Airlines**: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- **Telecommunication**: For keeping records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- **Web-based services**
    - **Social media**: For keeping records of users, connections between users (such as friend/follows information), posts made by users, rating/like information about posts, etc.
    - **Online retailers**: For keeping records of sales data and orders as for any retailer, but also for tracking a user's product views, search terms, etc., for the purpose of identifying the best items to recommend to that user.
    - **Online advertisements**: For keeping records of click history to enable targeted advertisements, product suggestions, news articles, etc. People access such databases every time they do a web search, make an online purchase, or access a social-networking site.
- **Document databases**: For maintaining collections of new articles, patents, and published
  research papers, etc.

- **Navigation systems**: For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, and other transportation options.

5. History of Database Management System

Techniques for data storage and processing have evolved over the years.

1. **1950s and early 1960s**: Magnetic tapes were developed for data storage. Data-processing tasks such as payroll were automated, with data stored on tapes. Processing of data consisted of reading data from one or more tapes and writing data to a new tape. For example, salary raises were processed by entering the raises on punched cards and reading the punched card deck in synchronization with a tape containing the master salary details. The records had to be in the same sorted order. The salary raises would be added to the salary read from the master tape and written to a new tape; the new tape would become the new master tape.

   In the early 1960s Integrated Data Store (IDS) was developed by Charles William Bacman and team. It is considered one of the early database management systems. It was based on the idea that data can point to other data. It was based on network model.
2. **Late 1960s and early 1970s**: Widespread use of hard disks in the late 1960s changed. the scenario for data processing greatly since hard disks allowed direct access to data. The position of data on disk was immaterial since any location on disk could be accessed in just tens of milliseconds. In June of 1970, Edgar Frank Codd, a British mathematician working as researcher in IBM published a research paper "**A Relational Model of Data for Large Shared Data Banks**". This paper lays the foundation for the development of the Relational Model.
3. **Late 1970s and 1980s**: Although academically interesting, the relational model was not used in practice initially because of its perceived performance disadvantages; relational databases could not match the performance of existing network and hierarchical databases. That changed with System R, a groundbreaking project at IBM Research that developed techniques for the construction of an efficient relational database system. In UC Barkley, two researchers started the implementation of the Codd's paper with a database system called Ingres.

   Larry Ellison and his two friends and former colleagues, Bob Miner and Ed Oates, started a consultancy called Software Development Laboratories (SDL) in 1977, which later became Oracle Corporation. SDL developed the original version of the Oracle software. In 1979, then SDL released a database management system

called Oracle based on the relational model. In response, IBM released their own version of a relational database management system called DB2 in 1983.

In 1986, the American National Standards Institute (ANSI) made Structured Query Language (SQL) the standard language to be used for Database Management System (DBMS)

4. **1990s**: In the early 1990s, decision support and querying re-emerged as a major application area for databases. Tools for analyzing large amounts of data saw a large growth in usage. Many database vendors introduced parallel database products in this period. Database vendors also began to add object-relational support to their databases.
   The major event of the 1990s was the explosive growth of the World Wide Web. Databases were deployed much more extensively than ever before. Database systems now had to support very high transaction-processing rates, as well as very high reliability and 24 × 7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities). Database systems also had to support web interfaces to data.

5. **2000s**: The types of data stored in database systems evolved rapidly during this period. Semi-structured data became increasingly important. XML emerged as a data-exchange standard. JSON, a more compact data-exchange format well-suited for storing objects from JavaScript or other programming languages, subsequently grew increasingly important. Increasingly, such data were stored in relational database systems as support for the XML and JSON formats was added to the major commercial systems. Spatial data (that is, data that includes geographic information) saw widespread use in navigation systems and advanced applications. Database systems added support for such data.
6. **2010s**: The variety of new data-intensive applications and the need for rapid development, particularly by startup firms, led to “NoSQL” systems that provide a lightweight form of data management. The name was derived from those systems’ lack of support for the ubiquitous database query language SQL, though the name is now often viewed as meaning “not only SQL.” In August of 2009, MongoDB version 1.0 was released.
7. **2020s**: In the 2020s, the rise of artificial intelligence (AI) and large language models (LLMs) revolutionized data management and retrieval. Traditional relational systems evolved alongside vector databases, which store and search high-dimensional embeddings that capture the semantic meaning of data rather than exact matches. This enabled powerful applications such as semantic search, context-aware retrieval, and AI-driven recommendations. Modern systems like Pinecone, FAISS, Weaviate, and ChromaDB emerged as key platforms for

managing these embeddings, while established databases such as PostgreSQL and MongoDB added native vector extensions. The decade represents a paradigm shift—databases are no longer just for storing structured facts, but for powering intelligent, meaning-based data access integrated with AI workflows.

6. File System Vs DBMS

A 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. Keeping organizational information in a file-processing system has a number of major disadvantages:

- **Data redundancy and inconsistency**

When different programmers create and manage separate files over time, data can become redundant and inconsistent. The same information may be stored in multiple places with different formats or programming languages. For instance, a student's contact details might appear in both the Music and Mathematics department files, causing duplication and higher storage costs. If one file is updated but others are not, the system ends up with conflicting (inconsistent) data across records.

- Difficulty in Accessing Data:

In traditional file-based systems, accessing specific information is difficult because data retrieval depends on prewritten programs. If a request—like listing students from a certain postal code—was not anticipated, a new program must be written or data must be filtered manually, both inefficient options. Each new query may require new code, making the process slow and inflexible. This highlights the need for **more flexible and efficient data-retrieval systems**, which later led to the development of database management systems (DBMS).

- Concurrent-access anomalies:

When multiple users update data at the same time, **concurrent-access anomalies** can occur, causing incorrect or inconsistent results. For example, if two bank clerks simultaneously withdraw money from the same account, both might read the same initial balance and overwrite each other's updates, leaving the final balance wrong. Similarly, two students registering for a course at the same time might both be accepted even if the class is already full, due to simultaneous updates of the registration count. These issues arise because concurrent programs can interfere with each other's operations, and without proper **supervision or transaction control**, data integrity cannot be guaranteed.

- **Security Problems**:

Security problems. Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

7. Characteristics of Database Approach

Key characteristics that define this approach include:

1. **Self-describing nature** – the database stores both data and metadata.
2. **Program-data independence** – programs are insulated from how data is physically stored.
3. **Multiple data views** – different users can see data in ways suited to their needs.
4. **Data sharing and multiuser transaction support** – many users can access and update data simultaneously without conflicts.

8. Data Abstraction and Independence

In a database system, **data abstraction** refers to the process of hiding the details of how data is stored and maintained, while presenting only essential information to the users. It allows users to interact with data without needing to understand complex internal details.

To achieve data abstraction, the database system is organized into **three levels**:

a) **Physical Level (Internal Level)**

- Describes **how the data is actually stored** in the database.
- It deals with data structures, file organization, indexing, and data paths.
- **Example:** Data may be stored as B+ trees, hash files, or sequential files on disk.

b) **Logical Level (Conceptual Level)**

- Describes **what data is stored** in the database and **the relationships among those data**.
- It defines the structure of the entire database using data models such as ER diagrams or relational schemas.
- **Example:** The database contains entities like *Student*, *Course*, and *Enrollment* with their attributes and relationships.

c) **View Level (External Level)**

- Describes **only a part of the database** that a particular user is interested in.

- Different users may have different views of the same database.
- **Example:** A student sees only their own grades, while an instructor sees grades for all students in a course.

Data independence refers to the ability to modify the database schema at one level of the system without affecting the schema at the next higher level. It ensures that changes in how data is stored or organized do not affect how users or applications access that data.

There are two types of data independence:

a. **Physical Data Independence**

The ability to change the physical schema (how data is stored) without changing the logical schema.

Example: Changing from a heap file to a B+ tree index should not affect the logical structure or applications.

b. **Logical Data Independence**

The ability to change the logical schema (conceptual design) without affecting the external schemas or application programs.

Example: Adding a new attribute or combining two relations should not require users or applications to change.

9. Application Architecture

Database applications can be partitioned into two or three parts, as shown in Figure 2. Earlier-generation database applications used a two-tier architecture, where the application resides at the client machine and invokes database system functionality at the server machine through query language statements.

In contrast, modern database applications use a three-tier architecture, where the client machine acts as merely a front end and does not contain any direct database calls; web browsers and mobile applications are the most commonly used application clients today. The front end communicates with an application server. The application server, in turn, communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications provide better security as well as better performance than two-tier applications.

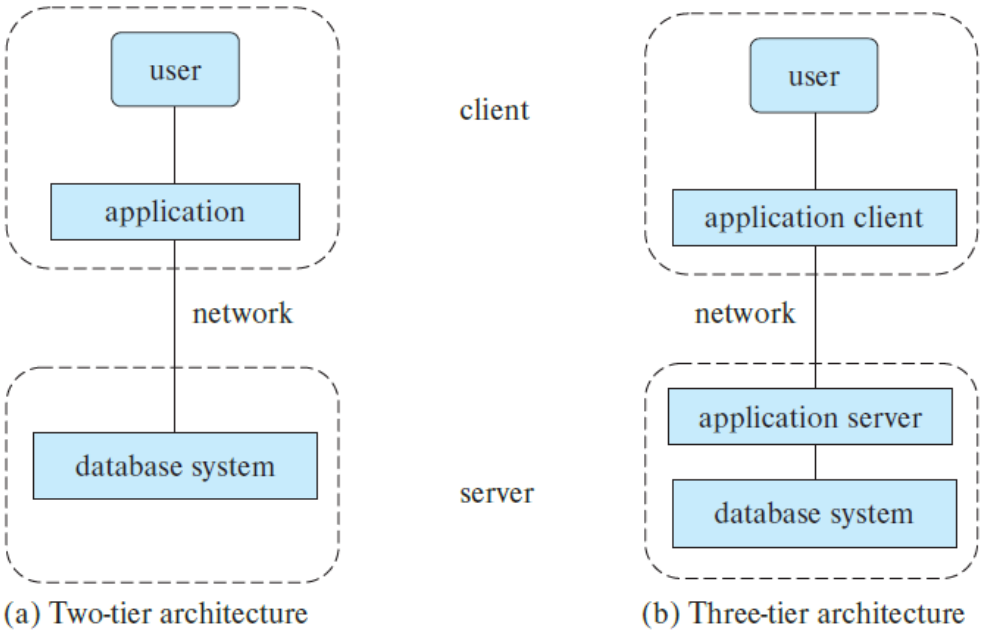

Figure 2: Two-tier and three-tier-architecture

10. Schemas and Instances

Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database. The overall design of the database is called the database schema. The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema.

11. Database Architecture

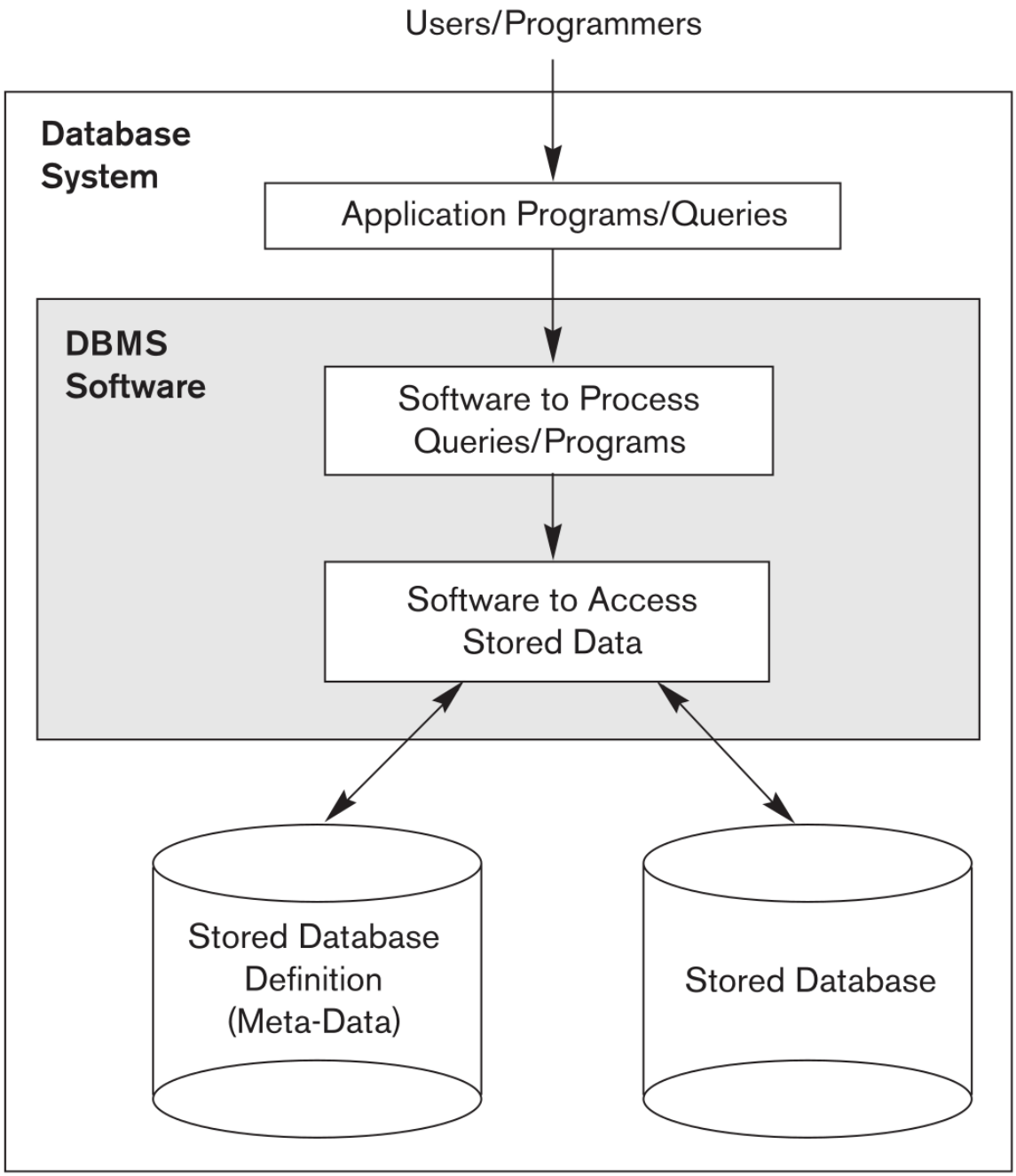

Figure 3: Database Architecture

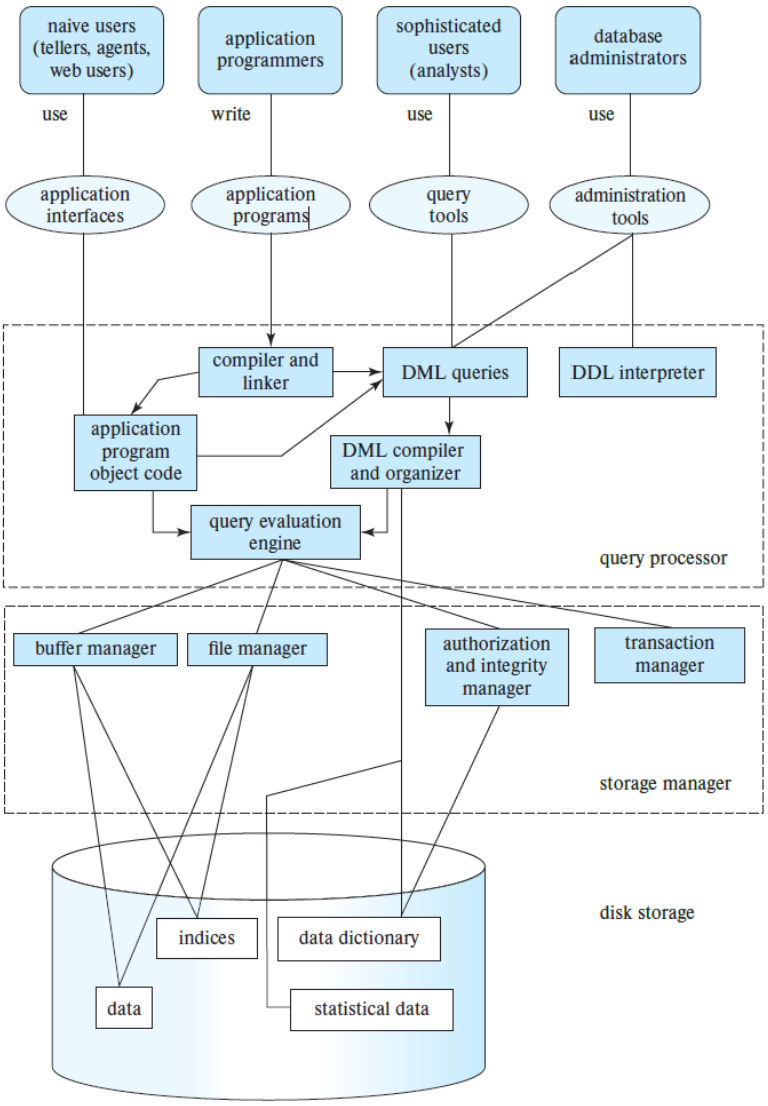

Figure 4: Database Architecture

12. Database Managers and Users

There are four types of database users, each interacting with the system differently depending on their role and technical expertise.

a) Naïve Users
- Use predefined user interfaces, such as web or mobile applications.
- Interact with the database through forms to enter or view data.
- Example: A student registering for a class using an online registration form.

b) Application Programmers

- computer professionals who write application programs to interact with the database.
- Use various programming tools and frameworks to build user interfaces and database applications.

c) Sophisticated Users
   - Do not write full application programs but use query languages (like SQL) or data analysis tools.
   - Typically include data analysts and scientists who query the database directly to analyze or explore data.

d) Database Administrator (DBA)

A Database Administrator (DBA) is responsible for central control of the data and the programs that access it. The DBA ensures data integrity, performance, and security of the database system.

Key Functions of the DBA

1. Schema Definition
   a. Creates the database schema using Data Definition Language (DDL) commands.
   b. Storage Structure and Access-Method Definition
   c. Specifies how data is physically stored and which indexes are used.

2. Schema and Physical Modification
   a. Updates schema or storage structures as organizational needs change or for performance improvement.
3. Authorization Management
   a. Grants and manages user permissions to control data access.
   b. Maintains an authorization structure used by the system during data access.

4. Routine Maintenance
   a. Database backup to prevent data loss.
   b. Disk space management to ensure smooth operations.

5. Performance monitoring to prevent overload from costly user queries.

References

Abraham Silberschatz, H. F. (2020). *Database System Concepts* (Seventh ed.). McGrawHillEducation.

Raghu Ramakrishnan, J. G. (n.d.). *Database management Systemts* (Second ed.). McGrawHill Education.

Ramez Elmasri, S. B. (2016). *Fundamentals of Database Systems* (Seventh ed.). Pearson.