SASTRA
DEEMED TO BE UNIVERSITY
THINK MERIT · THINK TRANSPARENCY · THINK SASTRA

INT104
Database Management Systems

B.Tech. (CSBS)

Unit I

Contents

- Introduction
  - Database-System Applications
  - Purpose of Database Systems
  - View of Database
    - Data Abstraction
    - Data Independence

Introduction

- 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*.

Database-System Applications

- Databases are widely used. Here are some representative applications:
- ***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 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.
  - *Online retailers*: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

- ***Banking and Finance***
  - *Banking*: For customer information, accounts, loans, and banking transactions.
  - *Credit card transactions*: For purchases on credit cards and 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 made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.*

Purpose of Database Systems

- Before the introduction of database systems, files were used to store and process the information.
- The purpose of database systems is to overcome the difficulties in file processing system
- Keeping organizational information in a file-processing system has a number of major disadvantages:
- **Data redundancy and inconsistency**.
  - **Data Redundancy:** Same information may be duplicated in several places (files).
    - For example, if a student has joined in two courses (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department.
    - This redundancy leads to higher storage and access cost.
    - In addition, it may lead to **data inconsistency**; that is, the various copies of the same data may not have updated data, if not properly updated in all files. For example, address changed in Music department records but not changed in mathematics department file.

- **Difficulty in accessing data**. Conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.
  - To get student details coming from 'Chennai' if there is no provision given in the file based application program, then we need to take entire student list and extract student details who are all coming from Chennai. Otherwise, a separate function to perform this task needs to be created and added the application program.
- **Data isolation**. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Integrity problems**. The data values stored in the database must satisfy certain types of **consistency constraints**.
  - In file processing system, the conditions needs to be checked in application programs. It is not possible to add constrains in the files.
  - If age of a student should be above 21, this condition should be checked in application program. It is not possible to check the condition in a file.

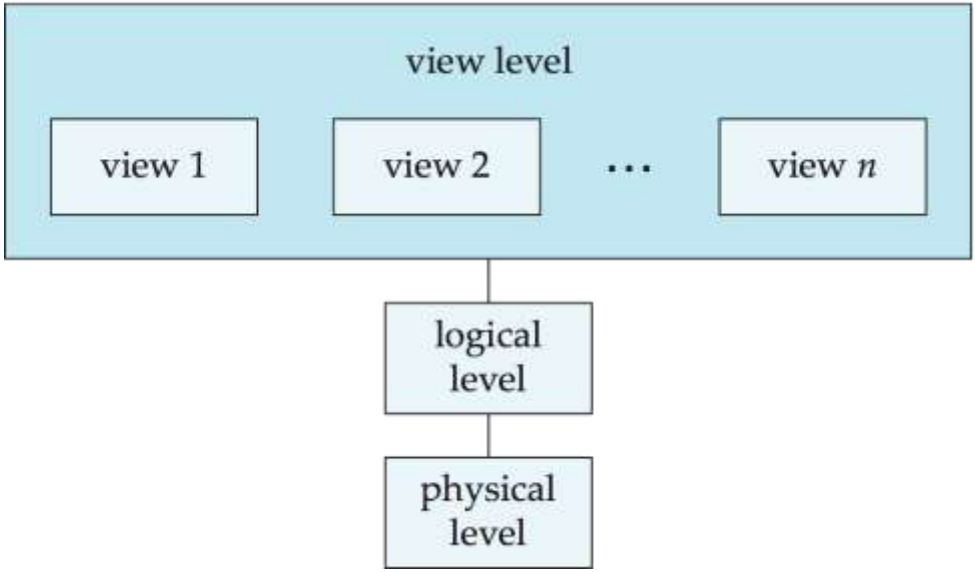
- **Atomicity problems**. "It must happen in its entirety or not at all". That is, in fund transfers, the money must be deducted in one account, and the same must be added in another account. In case of system failure, the transfer must be cancelled and actual balance amount should be maintained in both account.
  - It is difficult to ensure atomicity in a conventional file-processing system.
- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- **Security problems**
  - In file processing systems, it is difficult to restrict users to access only certain data.

Database systems offer solutions to all the above problems

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.
- **Data Abstraction**
  - To retrieve data efficiently, database systems uses complex data structures. These complexity details are hided from normal database users through several levels of abstraction, to simplify users' interaction with the systems.
  - **Physical level**. The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.

- **Logical level**
  - Describes *what* data are stored in the database, and what relationships exist among those data.
  - The user of the logical level does not need to be aware of the physical level complexity. This is referred to as **physical data independence**.
- **View level**.
  - Many users of the database system do not need complex data structures used in physical level and complex relationships described in logical level, instead, they need to access only a part of the database.
  - The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.


view level
view 1
view 2
...
view n
logical level
physical level

- For example, let us compare these concepts with structures in C language.

```
struct instructor {
        char ID [5];
        char name[20];
        char deptname[20];
        float salary;
}
```

- This code defines a new record type called *instructor* with four fields. Each field has a name and a type associated with it.

- A university organization may have several such record types, including
    - *department*, with fields *dept name*, *building*, and *budget*
    - *course*, with fields *course id*, *title*, *dept name*, and *credits*
    - *student*, with fields *ID*, *name*, *dept name*, and *tot cred*
- At the physical level, an *instructor*, *department*, or *student* record can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers.
- At the logical level, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well. Programmers using a programming language work at this level of abstraction. Similarly, database administrators usually work at this level of abstraction.
- Finally, at the view level, computer users see a set of application programs that hide details of the data types. At the view level, several views of the database are defined, and a database user sees some or all of these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database.

- **Instances and Schemas**
  - 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.

- **Data Model:**
  - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - A data model provides a way to describe the design of a database at the physical, logical, and view levels.
  - The data models can be classified into four different categories:

1. Relational model
   - Uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**.
2. Entity-Relationship data model (mainly for database design)
   - Uses a collection of basic objects, called *entities*, and *relationships* among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The entity-relationship model is widely used in database design.

3. Object-based data models (Object-oriented and Object-relational)
   - Extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.
4. Semi structured data model (XML)
   - This data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The **Extensible Markup Language (XML)** is widely used to represent semistructured data.
5. Other older models:
   - Network model
   - Hierarchical model

- **Database Languages**
  - A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and up dates.
  - **Data-Manipulation Language**
    - Enables users to access or manipulate data as organized by the appropriate data model. • Retrieval of information stored in the database
      - Insertion of new information into the database
      - Deletion of information from the database
      - Modification of information stored in the database
    - There are basically two types:
      - Procedural DMLs require a user to specify what data are needed and how to get those data.
      - Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.
    - A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **query language**.

- **Data-Definition Language**
  - Used to specify database schema by a set of definitions
  - Creating schema, modifying schema and deleting scheme
  - The data values stored in the database must satisfy certain **consistency constraints**.
  - The DDL provides facilities to meet the constraints with the help of integrity constrains
  - The database system checks these constraints every time the database is updated.

- **Database Users and Administrators**