جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
Department of Cyber Security

Subject:

Database

Class:

Second

Lecturer:

M.Sc. Ali Haider Alazam

M.Sc. Mohammad Baqer Haleem

Lecture: (1)

Introduction

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 systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

Database-System Applications

Database systems began in the 1960s to manage commercial data. Early systems handled simple, structured data, while modern databases manage large, valuable, and complex data used by many users at the same time. Today, many organizations are valuable mainly because of the data they own.

Traditional database applications, such as university systems, store well-structured and repetitive data. Modern applications, like social-networking platforms, store more complex and less structured data, including text, images, and user relationships.

Despite these differences, all database systems focus on managing data efficiently. To handle complexity, database systems use abstraction, which hides storage details and provides users with a simple view of the data. This allows organizations to store and manage different types of data in a unified system.

Some representative applications include:

- 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.
- 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.
- Universities: For student information, course registrations, and grades.
- Airlines: For reservations and schedule information.
- Telecommunications: 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

Database systems arose in response to early methods of computerized management of commercial data (File-Processing Systems). A file-processing system has several major disadvantages compared to a DBMS:

1. Data Redundancy : and Inconsistency: In file systems, different programmers create different files and application programs over a long period. This leads to:

- Redundancy: The same information (e.g., student address) is duplicated in several files.

- Inconsistency: Copies of the same data may not agree (e.g., an address change is updated in one file but not another.(

2. Difficulty in Accessing Data: File systems do not allow needed data to be retrieved in a convenient and efficient manner. New programs must be written for every new task.
3. Data Isolation: Data are scattered in various files, and files may be in different formats, making it difficult to write new application programs to retrieve appropriate data.
4. Integrity Problems: Data values must satisfy certain consistency constraints (e.g., account balance > 0). In file systems, these constraints are buried in program code, making them hard to add or change.
5. Atomicity Problems: A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure (e.g., a fund transfer must either complete entirely or not happen at all). This is hard to ensure in file systems.
6. Concurrent-Access Anomalies: For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Without supervision (which file systems lack), this can result in inconsistent data.
7. Security Problems: Not every user of the database system should be able to access all the data. Enforcing security constraints in an ad-hoc file system is difficult.

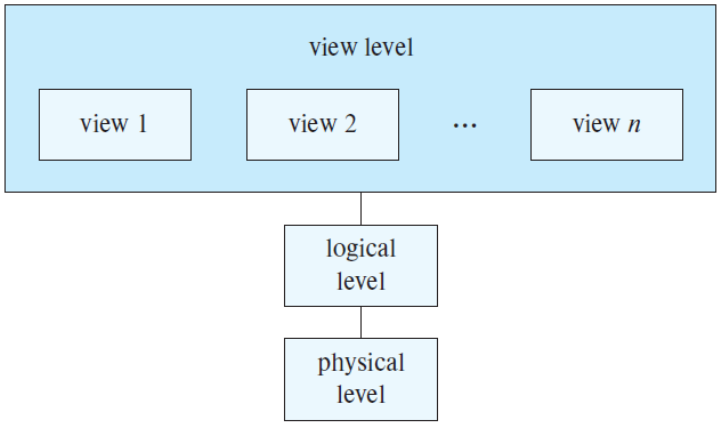
View of Data

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: For the system to be usable, it must retrieve data efficiently. The need for efficiency has led database system developers to use complex data structures to represent data in the database. Since many

database-system users are not computer trained, developers hide the complexity from users through several levels of data abstraction, **to simplify users' interactions with the system:**

- Physical Level: The lowest level of abstraction describes how the data are actually stored. It deals with complex low-level data structures.
- Logical Level: The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. Database administrators (DBAs) use this level.
- View Level: The highest level of abstraction describes only part of the entire database. The system may provide many views for the same database (e.g., a teller view vs. a manager view).



- Instances and Schemas:
  - Schema: The logical design of the database (analogous to type information of a variable in programming).
  - Instance: The collection of information stored in the database at a particular moment (analogous to the value of a variable).
  - Physical Data Independence: The ability to modify the physical schema without changing the logical schema.
- Data Models: A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

1. Relational Model: Uses a collection of tables to represent both data and the relationships among those data. It is the most widely used model.
2. Entity-Relationship (E-R) Model: Uses a collection of basic objects, called entities, and relationships among these objects.
3. Semi-structured Data Model: (e.g., XML, JSON) allows data specifications where individual data items of the same type may have different sets of attributes.
4. Object-Based Data Model: Extends the E-R model with notions of encapsulation, methods, and object identity.

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 updates.

- Data-Manipulation Language (DML): A language that enables users to access or manipulate data as organized by the appropriate data model.
  Types:
  - Procedural DMLs: Require a user to specify what data are needed and how to get those data.
  - Declarative DMLs (Non-procedural): Require a user to specify what data are needed without specifying how to get those data.
  - **Query Language:** The portion of a DML that involves information retrieval.
- Data-Definition Language (DDL): Used to specify the database schema. The DDL interpreter interprets DDL statements and records them in a set of tables containing metadata (data about data), called the **Data Dictionary** or **System Catalog.**
  - Integrity constraints (e.g., Domain constraints, Referential integrity) are specified in DDL.
  - Authorization (who can access what) is also specified here.

- SQL (Structured Query Language): The standard language for commercial relational database systems. It is not a Turing-complete language but is often embedded in host languages (C, Java, Python) for application development.

Database Design

Database design mainly involves the design of the database schema.

- Design Process:
  1- Characterize the data needs of the prospective database users.
  2- Conceptual Design: Choose a data model (often E-R model) and translate requirements into a conceptual schema.
  3- Logical Design: Map the conceptual schema to the implementation data model (e.g., Relational Tables).
  4- Physical Design: Specify the physical features of the database (file organization, indices).
- Normalization: A technique used in the design of relational database schemas to minimize redundancy and dependency.

Database Engine

The database system is partitioned into modules that deal with each of the responsibilities of the overall system.

- Storage Manager: A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. Components include:
  - Authorization and integrity manager.
  - Transaction manager.
  - File manager.
  - Buffer manager.
- Query Processor: Components include:

- DDL interpreter.
- DML compiler: Translates DML statements into an evaluation plan consisting of low-level instructions. It performs Query Optimization to pick the lowest cost evaluation plan.
- Query evaluation engine: Executes low-level instructions.

- Transaction Management:
  - Transaction: A collection of operations that performs a single logical function in a database application.
  - ACID Properties: Atomicity, Consistency, Isolation, Durability.
  - Concurrency-Control Manager: Controls the interaction among the concurrent transactions to ensure the consistency of the database.
  - Recovery Manager: Ensures that the database remains in a consistent (correct) state despite system failures.

Database Architecture

- Centralized Architecture: One server handles all processing.
- Client-Server Architecture:
  - Two-tier: Application resides at the client machine and invokes database system functionality at the server machine.
  - Three-tier: The client machine acts as merely a front end and does not contain any direct database calls. It communicates with an **application server** (business logic), which in turn communicates with the database system.
- Parallel Databases: Multiple processors and disks are used to improve processing speed (Speed-up and Scale-up).
- Distributed Databases: A collection of physically separate database systems, connected by a computer network.

Database Users and Administrators

- Database Users:
    1- Naive users: Unsophisticated users who interact with the system by invoking one of the application programs (e.g., a cashier).
    2- Application programmers: Computer professionals who write application programs.
    3- Sophisticated users: Interact with the system without writing programs, using a database query language or data analysis tools (e.g., analysts).
    4- Specialized users: Write specialized database applications (e.g., CAD systems, expert systems).
- Database Administrator (DBA): A person who has central control over the system. Functions include:
    1- Schema definition.
    2- Storage structure and access-method definition.
    3- Schema and physical-organization modification.
    4- Granting of authorization for data access.
    5- Routine maintenance (backups, performance monitoring).

History of Database Systems

- 1950s/60s: Magnetic tapes, Punched cards. Sequential access.
- • Late 1960s/70s: Hard disks (direct access). Network and Hierarchical models.
- Ted Codd defined the Relational Model (1970). IBM developed System R and SQL.
- 1980s: Relational databases became dominant (SQL standard). Object-oriented databases emerged.
- 1990s: Growth of the World Wide Web. Client-server and Three-tier architectures. Data warehousing and data mining.

- 2000s: XML, Autonomic computing.
- 2010s: Big Data, NoSQL databases, Cloud databases.