Data Collection and DBMS

- Introduction to file systems and databases
- What is a file system? Why do we need it?
- What is DBMS? Why is it important?
- Codd’s 12 Rules for RDBMS
- Comparison between File System and DBMS
- Summary and conclusion

Database

- A structured collection of data stored electronically
- Allows for easy retrieval, management, and updating of data
- Importance in real-world applications

- A structured collection of data stored electronically
- Allows for easy retrieval, management, and updating of data
- Importance in real-world applications

Importance in real-world applications

1.Banking and Finance

Transaction management, fraud detection, and customer account management.

2. Healthcare

Patient data management, electronic medical records (EMR), treatment tracking.

3. E-commerce

Product catalogs, customer data, and real-time inventory management.

4. Education

Student records, course management, and academic performance tracking.

5. Government

Citizen records, public services management, tax databases.

The Need for a Database

- Managing large amounts of data
- Ensuring data integrity and security
- Making data accessible and shareable
- Supporting data analytics and decision-making

File System Concepts

- A file system is a method used by operating systems to store, retrieve, and organize data.
- File systems store data as flat files in directories.
- Example: FAT32, NTFS, etc.

How a File System Works

- Data stored in files within folders (hierarchical structure)
- Each file has a path that includes the directory
- OS handles reading and writing operations
- Examples of file types: .txt, .docx, .jpg, etc.

Need for a File System

- To manage and store large volumes of data effectively
- Organizes data for easy retrieval
- Provides basic access control (read/write permissions)
- Allows for file sharing among users and applications

File System Limitations

- Poor data security and integrity
- Redundancy and inconsistency in data storage
- Lack of concurrent access and transactions
- No querying capabilities, limited scalability

File System vs. Database

File System:

- Storage is flat and unstructured
- No built-in query processing
- Limited control over data redundancy

Database (DBMS):

- Organized, structured storage
- Efficient data retrieval and manipulation
- Data integrity, concurrency control, and security features

Transition from File System to DBMS

- Why organizations shifted from file systems to DBMS:
- Improved data management
- Elimination of redundancy and inconsistency
- Increased security and data integrity

DBMS?

- A Database Management System (DBMS) is software that interacts with users and applications to capture and analyze data in a database.
- Popular DBMS software: MySQL, Oracle, PostgreSQL, SQL Server

Key Features of a DBMS

- Data abstraction and independence
- Efficient data access and manipulation
- Multiple views of data
- Security and authorization control
- Backup and recovery support
- Transaction management and concurrency control

Need for DBMS

- Organized data storage
- Reduced data redundancy
- Data integrity and consistency
- Efficient data querying and reporting
- Enhanced data security

Types of DBMS

- Hierarchical DBMS
- Network DBMS
- Relational DBMS (RDBMS)
- Object-oriented DBMS (OODBMS)

Introduction to Codd's 12 Rules

- In 1985, Edgar F. Codd, the father of relational databases, developed 12 rules to define what constitutes a true relational database system.
- These rules are fundamental for understanding the core principles of RDBMS, which are essential for database design and operation as discussed in Elmasri and Navathe's work.

Rule 1 - Information Rule

- All data must be logically represented as tables (relations), where data is stored as values in rows and columns.
- Each table represents an entity or a relationship
- Importance: This principle ensures that data in the database is uniformly structured, making it easy to query and manipulate.
- Example: Employee records stored in a table with columns like Employee_ID, Name, and Department.

Rule 2 - Guaranteed Access Rule

- Every atomic data element should be retrievable through a combination of the table name, the primary key of the row, and the column name.
- each data point has a unique, unambiguous path for retrieval.
- Importance: Simplifies data access and ensures no complex navigation to locate specific data.
- Example: In an "Employee" table, an employee's salary can be retrieved by referencing the table, primary key (Employee_ID), and the "Salary" column.

Rule 3 - Systematic Treatment of NULL Values

- NULL values must be treated systematically and uniformly across the database. NULL is used to represent missing, unknown, or inapplicable information.
- NULL values are default values that may arise in relational databases.
- Importance: Allows flexibility in data entry and processing without violating data integrity or creating ambiguity.
- Example: An employee record where the phone number is not available is represented with NULL in the "Phone" column.

Rule 4 - Dynamic Online Catalog Based on the Relational Model

- The database schema (catalog) should be stored within the system and accessible like any other table via SQL queries.
- This emphasize the importance of metadata stored in system catalogs for database self-management.
- Importance: Ensures that the schema can be queried and manipulated just like user data, making the system more flexible.
- Example: Querying a system catalog to find details about the structure of the "Employee" table.

Rule 5 - Comprehensive Data Sublanguage Rule

- The database must support at least one comprehensive language that handles data definition, data manipulation, constraints, and transaction control (e.g., SQL).
- Importance: Provides users with a unified way to interact with the database across all aspects of its functionality.
- Example: SQL is used to create tables (CREATE TABLE), insert data (INSERT), query data (SELECT), and update data (UPDATE).

Rule 6 - View Updating Rule

- All views (virtual tables) that are theoretically updateable should be updateable in the database system.
- Importance: Ensures that users can work with virtual tables just as they do with base tables, improving flexibility.
- Example: A view combining employee names and departments should allow updating of either name or department directly.

Rule 7 - High-Level Insert, Update, and Delete

- The system must support set-oriented, high-level operations for inserting, updating, and deleting data.
- Importance: Provides powerful tools for managing data at scale, allowing batch operations on multiple rows.
- Example: SQL allows users to delete all employees from a specific department in one command: DELETE FROM Employee WHERE Department = 'Sales';.

- Slide 9: Rule 8 - Physical Data Independence
- Description:
- Changes in the physical storage of data should not affect how data is logically stored or accessed.
- This principle aligns with data independence as discussed in Elmasri & Navathe, where the logical schema is separated from the physical schema.
- Importance: Ensures that system optimization at the storage level does not impact users' ability to query and work with data.
- Example: Moving the database files to another disk should not affect SQL queries or the table structure.

Rule 9 - Logical Data Independence

- Changes to the logical structure of the database (e.g., adding new fields or tables) should not require changes in the applications that access the data.
- Importance: Allows flexibility in modifying the database without disrupting business operations.
- Example: Adding a new "Email" column to the "Employee" table should not break existing queries that do not use this field.

Rule 10 - Integrity Independence

- Integrity constraints must be defined and stored in the relational schema and must be independent of application programs.
- Importance: Ensures that data integrity rules (e.g., uniqueness, referential integrity) are enforced universally across the system.
- Example: Defining a primary key on the "Employee_ID" ensures uniqueness across all employee records.

Rule 11 - Distribution Independence

- The distribution of the database across various locations should be invisible to users. Whether data is stored locally or distributed over a network should not affect queries.
- Importance: Users should not need to worry about where data is physically located when querying or updating it.
- Example: Whether an "Employee" table is stored in New York or Tokyo, a user in London should be able to query it seamlessly.

Rule 12 - Non-Subversion Rule

- If a low-level (physical) access method exists, it should not bypass integrity constraints or security defined at the higher levels.
- Authorization and security controls should ensure that no unauthorized access is possible, even through low-level interfaces.
- Importance: Prevents users from bypassing relational integrity and security by accessing the database through non-relational means.
- Example: Even if a user accesses a database file directly, they should not be able to violate a foreign key constraint.