DBMS
Module-1

Dr. Hari Seetha

Types of Databases and Database Applications

- Numeric and Textual Databases
- Big Data Storage Systems or NoSQL databases
- Multimedia Databases
- Geographic Information Systems (GIS)
- Data Warehouses and OLAP systems
- Real-time and Active Databases

Basic Definitions

- **Database**: A collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Mini-world**: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS)**: *general-purpose software system* that facilitates the processes of *defining, constructing, manipulating,* and *sharing* databases among various users and applications.
- **Database System**: The DBMS software together with the data itself. Sometimes, the applications are also included.

Typical DBMS Functionality

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent

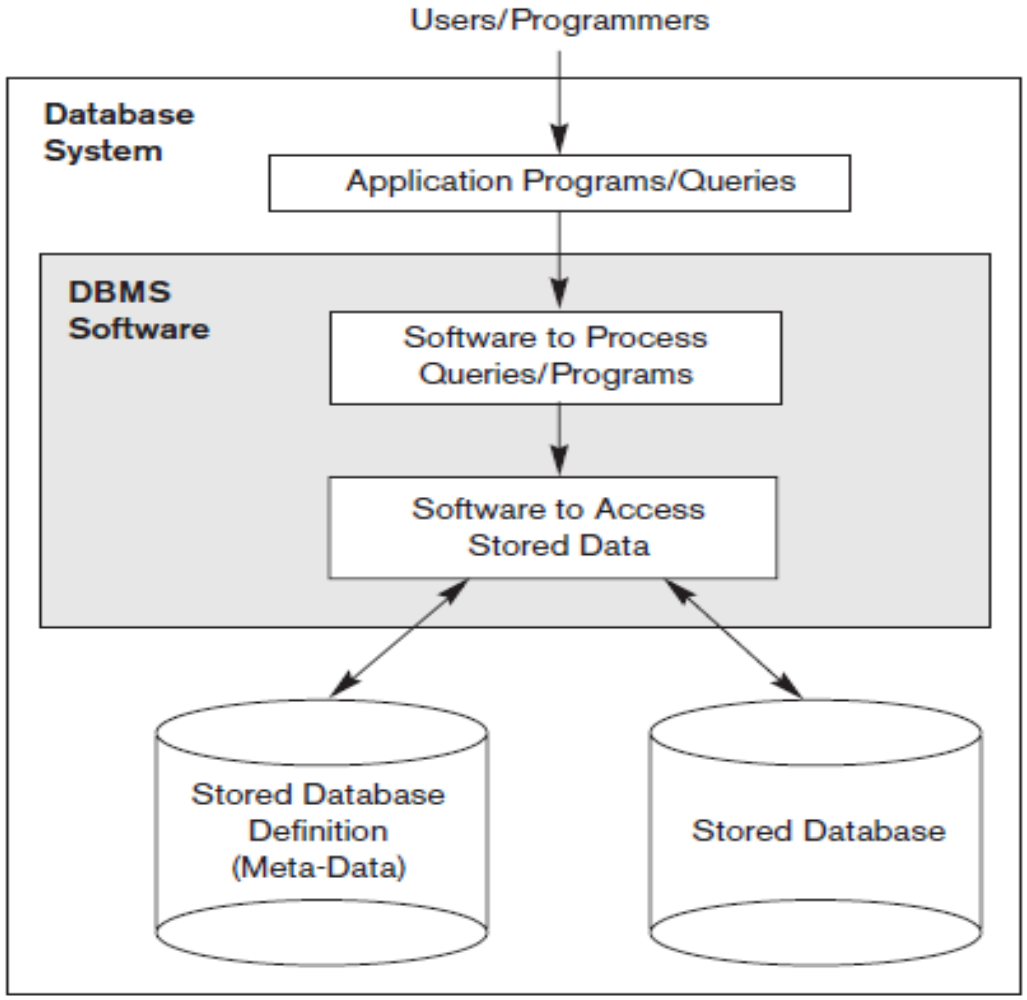
Typical DBMS Functionality

Other features:

- Protection or Security measures to prevent unauthorized access
- "Active" processing to take internal actions on data
- Presentation and Visualization of data

- An **application program** accesses the database by sending queries or requests for data to the DBMS.
- A **query** typically causes some data to be retrieved
- A **transaction** may cause some data to be read and some data to be written into the database.

A simplified database environment



Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example**: Part of a UNIVERSITY environment.
- **Some mini-world *entities***:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Note: The above could be expressed in the ENTITY-RELATIONSHIP data model.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships***:
  - SECTIONs *are of* specific COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have* prerequisite COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

Note: The above could be expressed in the *ENTITY-RELATIONSHIP* data model.

File System vs Database System

- Usage :File system helps to store a collection of raw data files into the hard disk. DBMS helps to easily store, retrieve and manipulate data in a database.
- In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application
- This is the main difference between file system and DBMS.

- Operations: Tasks such as storing, retrieving and searching are done manually in a file system. Therefore, it is difficult to manage data using a file system.

On the other hand, operations such as updating, searching, selecting data is easier in DBMS because it allows using SQL querying.

- Data Consistency: File system has data inconsistency whereas DBMS provides higher data consistency using normalization.

- Data Redundancy: There is more redundant data in a file system whereas there is low data redundancy in a DBMS.
- Security: DBMS provides more security to the data than the file system.
- Backup and Recovery Process: Backup and recovery process is not efficient in file system because it is not possible to recover the lost data. On the contrary, a DBMS has a sophisticated backup and recovery

- Users: File system is appropriate to handle data of a small-scale organization or individual users. On the other hand, DBMS is suitable for medium to large organizations or multiple users.
- Complexity: Handling the file system is simple but handling a DBMS is complex
- Examples: NTFS and Ext are some examples of file systems. MySQL, MSSQL, Oracle, and DB2 are some examples of DBMS.

Note: New Technology File System (NTFS) is the Windows file system. Extended File System (Ext) is the Linux file system.

Main Characteristics of the Database Approach

- Self-describing nature of a database system: A DBMS **catalog** stores the *description* of the database. (The description is called **meta-data**). This allows the DBMS software to work with different databases.
- Insulation between programs and data: Called **program-data independence**.
- Allows changing data storage structures and operations without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Support of multiple views of the data: Each user may see a different view of the database, which describes *only* the data of interest to that user.

Main Characteristics of the Database Approach

- **Sharing of data and multiuser transaction processing** : allowing a set of concurrent users to retrieve and to update the database.
- Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted.
- OLTP (Online Transaction Processing) is a major part of database applications.

Database Users

Users may be divided into

- Actors on the Scene -those who actually use and control the content

and

- "Workers Behind the Scene"-those who enable the database to be developed and the DBMS software to be designed and implemented.

Database Users

Actors on the scene

- **Database administrators:** responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database Designers:** responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.
- **End-users:** they use the data for queries, reports and some of them actually update the database content.

Workers behind Scene

■ **DBMS system designers and implementers** design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many **modules**, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security.

The DBMS must interface with other system software, such as the operating system and compilers for various programming languages.

■ **Tool developers** design and implement **tools**—the software packages that facilitate database modelling and design, database system design, and improved performance.

Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation. In many cases, independent software vendors develop and market these tools.

■ **Operators and maintenance personnel** (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system

Categories of End-users

- **Casual** : access database occasionally when needed
- **Naïve or Parametric** : they make up a large section of the end-user population.
- They use previously well-defined functions in the form of "canned transactions" against the database.
- Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Categories of End-users

- **Sophisticated** : these include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
- Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone** : mostly maintain personal databases using ready-to-use packaged applications.
- An example the user of a financial software package that stores a variety of personal financial data

Advantages of Using the Database Approach

- **Controlling redundancy in data storage and in development and maintenance efforts.**
  - Redundancy leads to duplication of effort, wastage of storage space and the files may become inconsistent
  - **data normalization in db approach** ensures consistency and saves storage space
  - in practice, it is sometimes necessary to use **controlled redundancy** to improve the performance of queries by placing all data together using **denormalization**
- **Sharing of data among multiple users.**
- **Restricting unauthorized access to data.**
  - DBMS provides a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions
  - controls the type of access operation—retrieval or update
- **Providing persistent storage for program Objects**
  - a complex object in C++ can be stored permanently in an object-oriented DBMS. Such an object is said to be **persistent**, since it survives the termination of program execution and can later be directly retrieved by another program.
  - Object-oriented database systems typically offer data structure **compatibility** with one or more object-oriented programming languages

Advantages of Using the Database Approach Contd.

- **Providing Storage Structures for efficient Query Processing**
  - Auxiliary files called **indexes** that speed up disk search are used for this purpose.
  -the DBMS often has a **buffering** or **caching** module that maintains parts of the database in main memory buffers
  -The **query processing and optimization** module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures.
- **Providing backup and recovery services.**
  - **backup and recovery subsystem** of the DBMS is responsible for recovery from hardware and software failures
  - Disk backup is also necessary in case of a catastrophic disk failure.
- **Providing multiple interfaces to different classes of users.**
  - apps for mobile users, query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users,and menu-driven interfaces and natural language interfaces for standalone users

Advantages of Using the Database Approach Contd.

- **Representing complex relationships among data.**
- **Enforcing integrity constraints on the database.**

-The simplest type of integrity constraint involves specifying a data type for each data item.

Uniqueness, Regerential Integrity etc

- **Drawing Inferences and Actions using rules**

-Some database systems provide capabilities for defining *deduction rules* for *inferencing* new information from the stored database facts. Such systems are called **deductive database systems**.

-RDBMS uses Triggers. A trigger is a form of a rule activated by updates to the table

-**active database systems**, provide active rules that can automatically initiate actions when certain events and conditions occur

Additional Implications of Using the Database Approach

- **Potential for enforcing standards:** this is very crucial for the success of database applications in large organizations Standards refer to data item names, display formats, screens, report structures, meta-data (description of data) etc.
- **Reduced application development time**: incremental time to add each new application is reduced.

Additional Implications of Using the Database Approach

- Flexibility to change data structures: database structure may evolve as new requirements are defined.
- Availability of up-to-date information – very important for on-line transaction systems such as airline, hotel, car reservations.
- Economies of scale: by consolidating data and applications across departments wasteful overlap of resources and personnel can be avoided.

Historical Development of Database Technology

- **Early Database Applications:** The Hierarchical and Network Models were introduced in mid 1960's and dominated during the seventies. A bulk of the worldwide database processing still occurs using these models.
- **Relational Model based Systems:** The model that was originally introduced in 1970 was heavily researched and experimented with in IBM and the universities. Relational DBMS Products emerged in the 1980's.

Historical Development of Database Technology

- **Object-oriented applications:** OODBMSs were introduced in late 1980's and early 1990's to cater to the need of complex data processing in CAD and other applications. Their use has not taken off much.
- **Data on the Web and E-commerce Applications:** Web contains data in HTML (Hypertext markup language) with links among pages. This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).

Extending Database Capabilities

- **New functionality is being added to DBMSs in the following areas:**
  - Scientific Applications
  - Image Storage and Management
  - Audio and Video data management
  - Data Mining
  - Spatial data management
  - Time Series and Historical Data Management

The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

When not to use a DBMS

- **Main inhibitors (costs) of using a DBMS**:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- **When a DBMS may be unnecessary:**
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.

When not to use a DBMS

- **When no DBMS may suffice:**
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.