Databases and Database Users

Part 2

Actors on the Scene

Database Administrators

- The **database administrator** (**DBA**) is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.
- The DBA is accountable for problems such as security breaches and poor system response time.

Database Designers

- **Database designers** are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.
- Database designers typically interact with each potential group of users and develop **views** of the database that meet the data and processing requirements of these groups.
- Each view is then analyzed and *integrated* with the views of other user groups.
- The final database design must be capable of supporting the requirements of all user groups.

End Users

- **End users** are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use.

- There are several categories of end users:
  - **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.
  - **Naive** or **parametric end users** make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called **canned transactions**—that have been carefully programmed and tested.
  - **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.
  - **Standalone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces.

System Analysts and Application Programmers (Software Engineers)

- **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements.
- **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions.

Workers behind the Scene

- **DBMS system designers and implementers** design and implement the DBMS modules and interfaces as a software package.
- **Tool developers** design and implement **tools**—the software packages that facilitate database modeling and design, database system design, and improved performance.
- **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.

Advantages of Using the DBMS Approach

Controlling Redundancy

- In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications.
- This **redundancy** in storing the same data multiple times leads to several problems.
  - First, there is the need to perform a single logical update—such as entering data on a new student—multiple times: once for each file where student data is recorded. This leads to *duplication of effort*.
  - Second, *storage space is wasted* when the same data is stored repeatedly, and this problem may be serious for large databases.
  - Third, files that represent the same data may become *inconsistent*.

- In the database approach, the views of different user groups are integrated during database design.
- Ideally, we should have a database design that stores each logical data item—such as a student’s name or birth date—in *only one place* in the database.
- This is known as **data normalization**, and it ensures consistency and saves storage.
- However, in practice, it is sometimes necessary to use **controlled redundancy** to improve the performance of queries.

GRADE_REPORT

(a)

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

GRADE_REPORT

(b)

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Brown	112	MATH2410	B

Restricting Unauthorized Access

- A DBMS should provide a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions.

Providing Persistent Storage for Program Objects

- Databases can be used to provide **persistent storage** for program objects and data structures.
- This is one of the main reasons for **object-oriented database systems**.
- Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.
- Hence, a complex object in C++ can be stored permanently in an object-oriented DBMS.

Providing Storage Structures and Search Techniques for Efficient Query Processing

- Database systems must provide capabilities for *efficiently executing queries and updates*.
- Because the database is typically stored on disk, the DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records.
- Auxiliary files called **indexes** are used for this purpose.
- In order to process the database records needed by a particular query, those records must be copied from disk to main memory.
- Therefore, 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.
- The choice of which indexes to create and maintain is part of *physical database design and tuning*, which is one of the responsibilities of the DBA staff.

Providing Backup and Recovery

- The **backup and recovery subsystem** of the DBMS is responsible for recovery.

Providing Multiple User Interfaces

- Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.

Representing Complex Relationships among Data

- A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

Enforcing Integrity Constraints

- Most database applications have certain **integrity constraints** that must hold for the data.
- A DBMS should provide capabilities for defining and enforcing these constraints.

Permitting Inferencing 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**.

- In today’s relational database systems, it is possible to associate **triggers** with tables.
- More involved procedures to enforce rules are popularly called **stored procedures**; they become a part of the overall database definition and are invoked appropriately when certain conditions are met.
- More powerful functionality is provided by **active database systems**, which 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
- Reduced Application Development Time
- Flexibility
- Availability of Up-to-Date Information
- Economies of Scale

When Not to Use a DBMS

- The overhead costs of using a DBMS are due to the following:
  - High initial investment in hardware, software, and training
  - The generality that a DBMS provides for defining and processing data
  - Overhead for providing security, concurrency control, recovery, and integrity functions

- Therefore, it may be more desirable to use regular files under the following circumstances:
  - Simple, well-defined database applications that are not expected to change at all
  - Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
  - No multiple-user access to data