Introduction to DBMS

Course Overview

Module No.	Content	Teaching Hours
I	Introduction: An Overview of Database Management System, Database System Vs File System, Database System Concept and Architecture, Data Model Schema and Instances, Data Independence, Database Language and Interfaces (DDL, DML, DCL), Database Development Life Cycle (DDLC) with Case Studies. Data Modeling Using the Entity-Relationship Model: ER Model Concepts, Notation for ER Diagram, Mapping Constraints, Keys, Specialization, Generalization, Aggregation, Reduction of an ER Diagram to Tables, Extended ER Model. Relational Data Model and Language: Relational Data Model Concepts, Integrity Constraints, Entity Integrity, Referential Integrity, Keys Constraints, Domain Constraints, Relational Algebra **Database Design & Normalization I:** Functional Dependencies, Primary Key, Foreign Key, Candidate Key, Super Key, Normal Forms, First, Second, Third Normal Forms, BCNF, Non-Redundant Cover, Canonical Cover	20

Course Overview

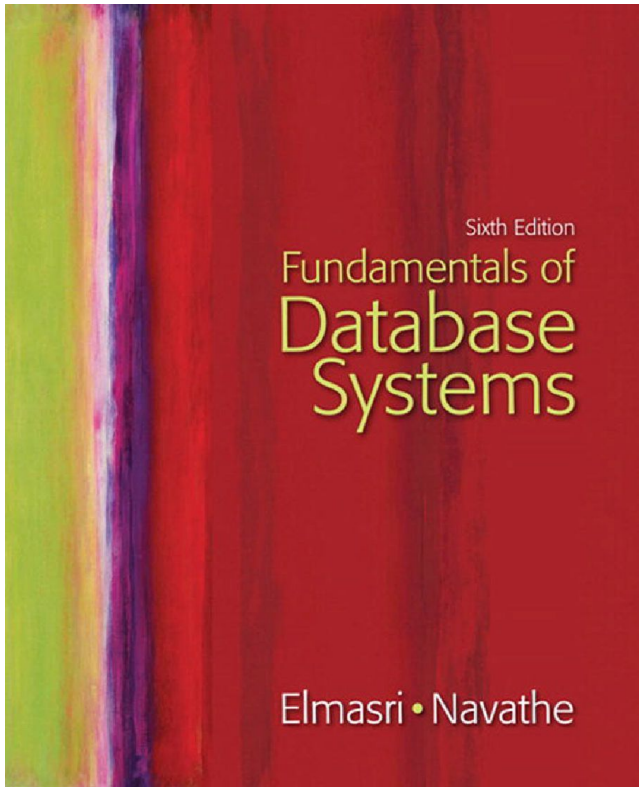
BCSC0802: DATABASE MANAGEMENT SYSTEM LAB

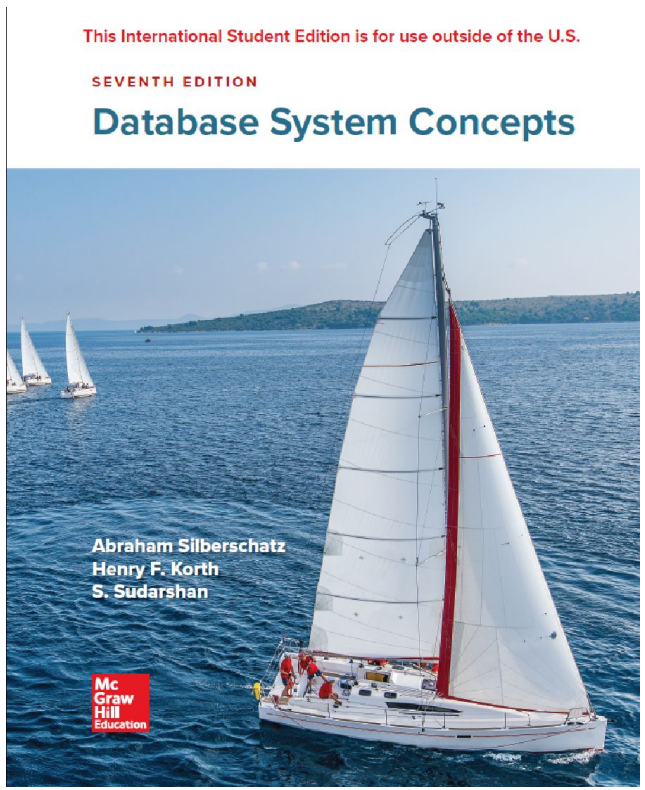
Objective: *The lab aims to develop understanding of different applications and constructs of SQL PL/SQL.*

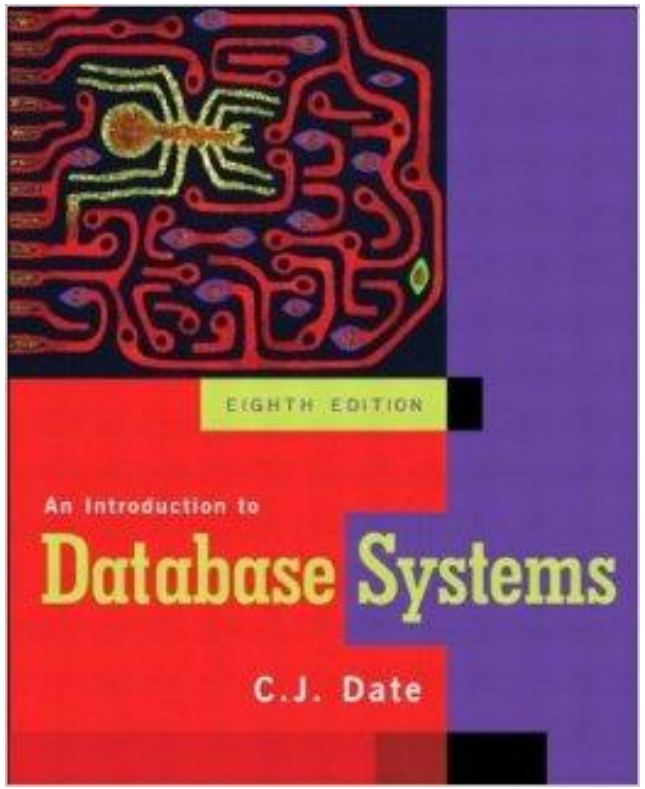
Credits:1 **L-T-P-J:0-0-2-0**

Module No.	Content	Teaching Hours
I, II and III	• Write the SQL queries for data definition and data manipulation language. • To implement various operations on a table. • To implement various functions in SQL. • To implement restrictions on the table. • To implement concept of grouping of Data. • To implement concept of Joins in SQL. • To implement the concept of sub-queries. • To implement the concept of views, sequence. • To implement the concept of PL/SQL using cursor. • To implement the concept of Procedure function and Triggers.	24

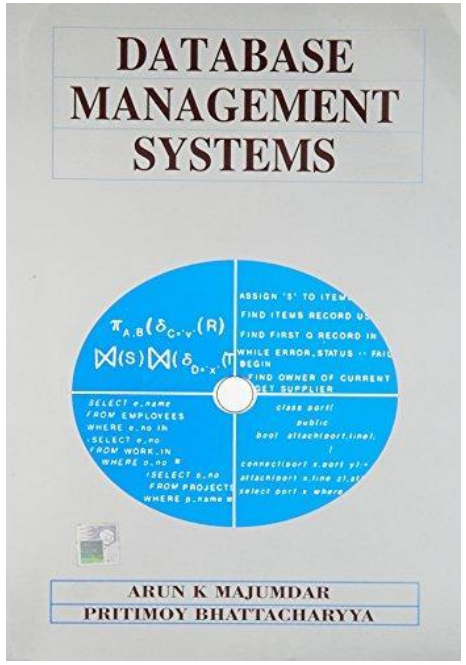
Literature

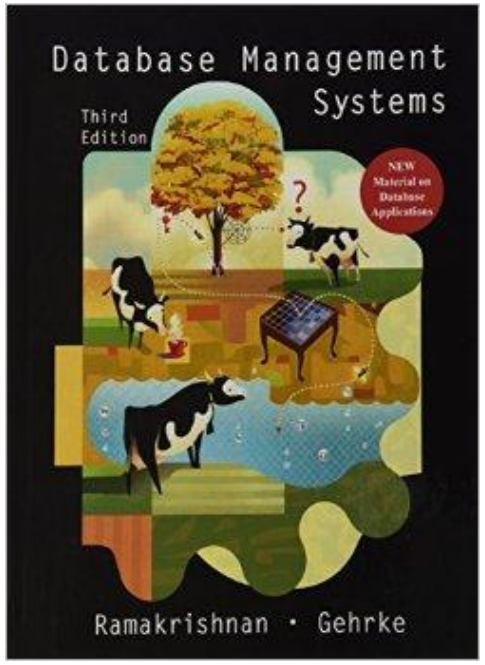

Sixth Edition
Fundamentals of
Database
Systems
Elmasri • Navathe

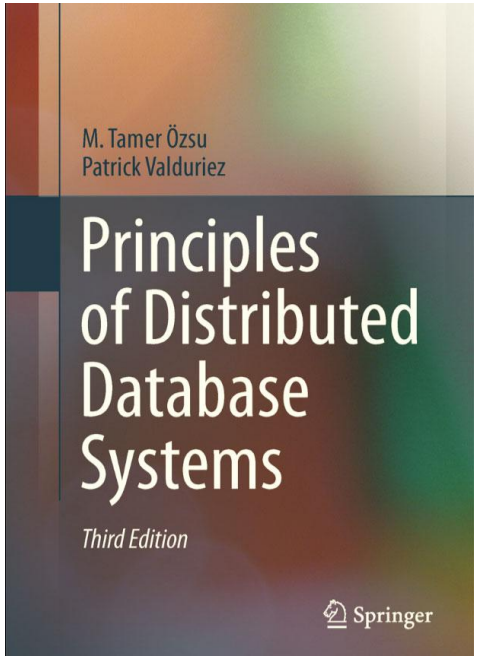

This International Student Edition is for use outside of the U.S.
SEVENTH EDITION
Database System Concepts
Abraham Silberschatz
Henry F. Korth
S. Sudarshan
Mc
Graw
Hill
Education


EIGHTH EDITION
An Introduction to
Database Systems
C.J. Date

Literature


DATABASE
MANAGEMENT
SYSTEMS
ARUN K MAJUMDAR
PRITIMOY BHATTACHARYYA


Database Management
Systems
Third
Edition
NEW
Material on
Database
Applications
Ramakrishnan · Gehrke


M. Tamer Özsu
Patrick Valduriez
Principles
of Distributed
Database
Systems
Third Edition
Springer

Data, Database, DBMS

- **Data:** Known facts that can be recorded and have an implicit meaning; raw
- **Database:** A highly organized, interrelated, and structured set of data about a particular enterprise
  - Controlled by a database management system (DBMS)
- **DBMS:**
  - A collection of programs that enables users to create and maintain a database
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
  - A software package/system to facilitate the creation and maintenance of a computerized database
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

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 systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.

- A modern database system is a complex software system whose task is to manage a large, complex collection of data.

Types of Databases and Database Applications

- Traditional applications:
  - Numeric and textual databases
- More recent applications:
  - Multimedia databases
  - Geographic Information Systems (GIS)
  - Biological and genome databases
  - Data warehouses
  - Mobile databases
  - Real-time and active databases

Recent Developments

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
  - Facebook
  - Twitter
  - Linked-In
- All of the above constitutes data
- Search Engines, Google, Bing, Yahoo: collect their own repository of web pages for searching purposes
- New technologies are emerging from the so-called non-SQL, non-database software vendors to manage vast amounts of data generated on the web:
  - **Big data** storage systems involving large clusters of distributed computers
  - **NOSQL** (Non-SQL, Not Only SQL) systems
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

What a DBMS Facilitates

- DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.
- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications (e.g., GLA Module)
- *Processing* and *sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

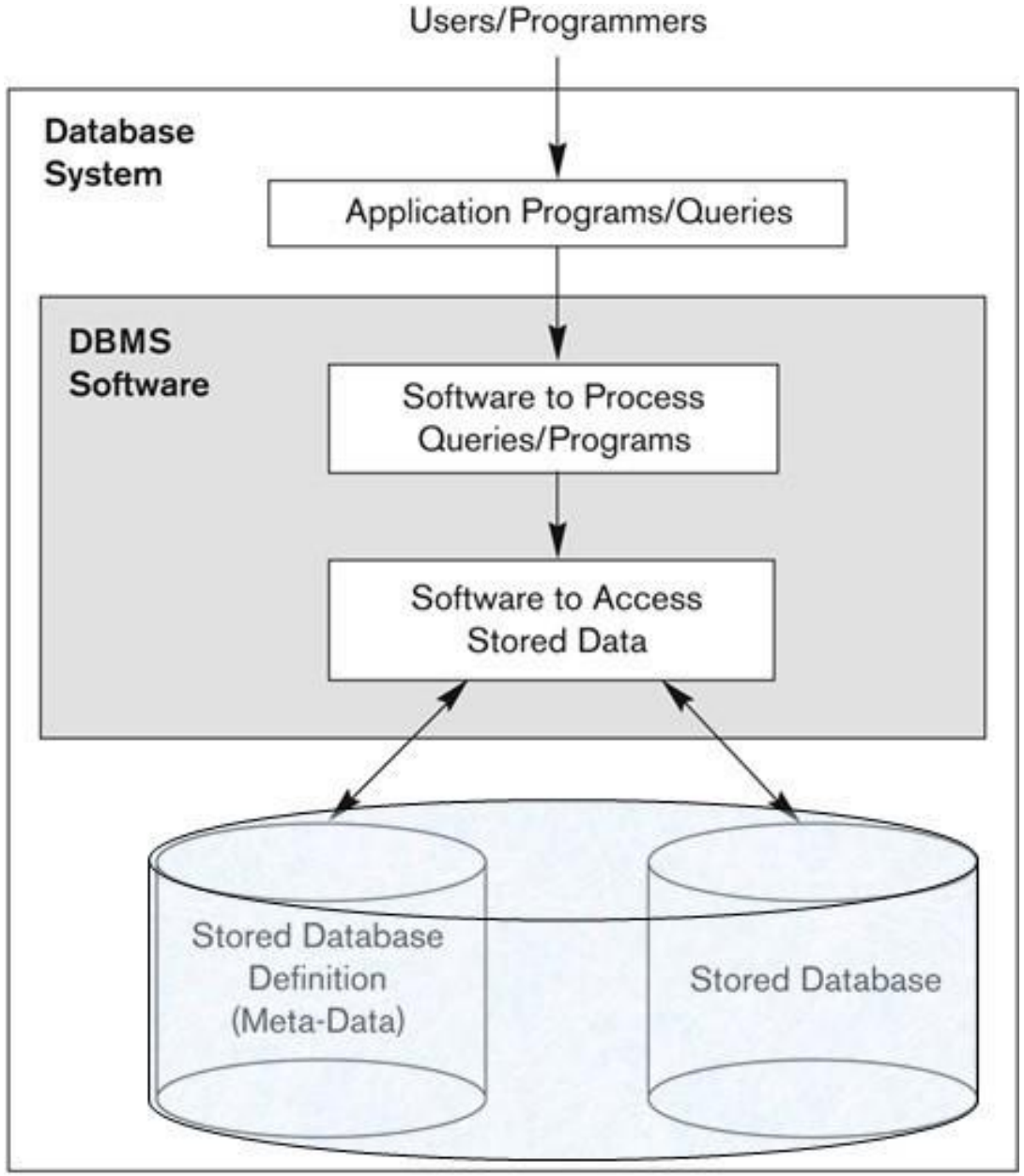
Other DBMS Functionalities

- DBMS may additionally provide:
  - Protection or security measures to prevent unauthorized access
  - Presentation and visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application

Application Programs and DBMS

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and “update” certain values or generate new data and store that in the database

Simplified database system environment



Example of a Database

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

Example of a Database

- 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
- The above entities and relationships are typically expressed in a conceptual data model, such as the entity-relationship (ER) data or UML class model.

Example of a Database

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

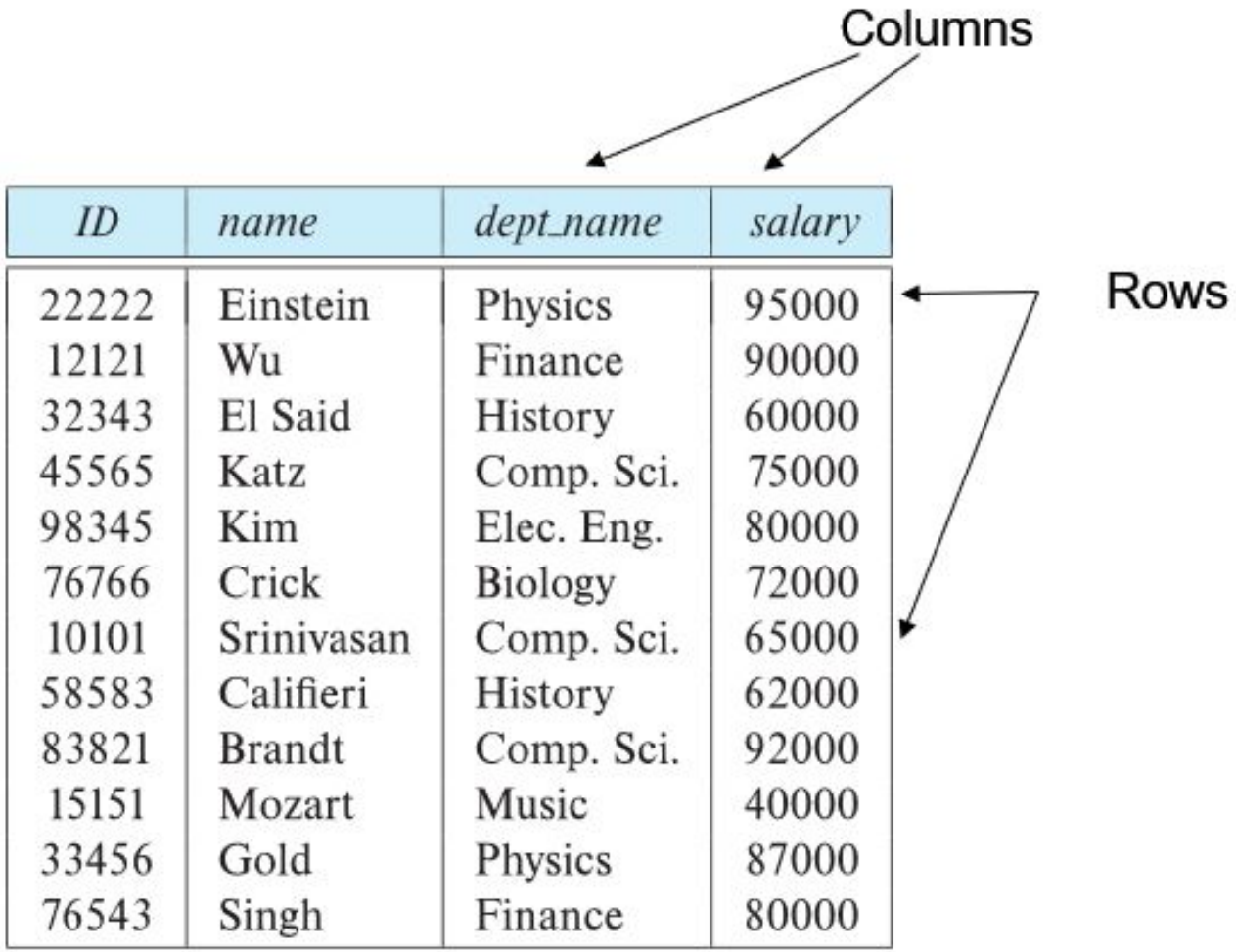
GRADE_REPORT

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

The relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Main Characteristics of the Database Approach

- A number of characteristics distinguish the database approach from the much older approach of programming with files. In traditional **file processing**, each user defines and implements the files needed for a specific software application as part of programming the application. For example, one user, the *grade reporting office,* may keep files on students and their grades. Programs to print a student's transcript and to enter new grades are implemented as part of the application. A second user, the *accounting office*, may keep track of students' fees and their payments. Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files. This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common up-to-date data.

- In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications.

Main Characteristics of the Database Approach

- The main characteristics of the database approach versus the file-processing approach are the following:
  - Self-describing nature of a database system
  - Insulation between programs and data, and data abstraction
  - Support of multiple views of the data
  - Sharing of data and multiuser transaction processing

Main Characteristics of the Database Approach

- Self-describing nature of a database system
  - Database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
  - This definition is stored in the DBMS **catalog**, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints).
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.
  - In traditional file processing, data definition is typically part of the application programs themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. For example, an application program written in C++ may have struct or class declarations. Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and using these definitions.

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Main Characteristics of the Database Approach (continued)

- Insulation between programs and data
  - In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file.
  - By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs.
  - This property is called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Main Characteristics of the Database Approach (continued)

- For example, a file access program may be written in such a way that it can access only STUDENT records of the structure as shown below:

Data Item Name	Starting Position in Record	Length in Characters (bytes)
Name	1	30
Student_number	31	4
Class	35	1
Major	36	4

- If we want to add another piece of data to each STUDENT record, say the Birth_date, such a program will no longer work and must be changed. By contrast, in a DBMS environment, we only need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birth_date; no programs are changed. The next time a DBMS program refers to the catalog, the new structure of STUDENT records will be accessed and used.

Main Characteristics of the Database Approach (continued)

❑ Data Abstraction

- The characteristic that allows program-data independence is called data abstraction.
- A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented.
- A data model is a type of data abstraction that is used to provide this conceptual representation.
- Data model hides storage and implementation details that are not of interest to most database users.
- A data model is used to hide storage details and present the users with a conceptual view of the database.
- The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts.
- Programs refer to the data model constructs rather than data storage details.

Main Characteristics of the Database Approach (continued)

- Support of multiple views of the data
  - A database typically has many users, each of whom may require a different perspective or view of the database.
  - A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.
  - 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 (continued)

❑ Sharing of data and multi-user transaction processing

- Allowing a set of concurrent users to retrieve from and to update the database.
- A multiuser DBMS must allow multiple users to access the database at the same time.
- DBMS must include **concurrency control** software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.
- Concurrency control within the DBMS guarantees that each transaction is correctly executed or aborted.
- For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger.
- These types of applications are generally called online transaction processing (OLTP) applications.
- OLTP is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.
- A **transaction** is an executing program or process that includes one or more database accesses, such as reading or updating of database records.
- Recovery subsystem ensures each completed transaction has its effect permanently recorded in the database.

Database Users

❑ Database administrators

- In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources.
- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.
- Database administrator (DBA) is responsible for authorizing access to the database, coordinating and monitoring its use, acquiring software and hardware resources as needed, controlling its use and monitoring efficiency of operations.
- 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.
- 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.

Database Users

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.
- They use the data for queries, reports and some of them update the database content.

System Analysts and Application Programmers

- **System analysts:** System analysts determine the requirements of end users and develop specifications for standard transactions that meet these requirements.
- **Application programmers:** Application programmers implement these specifications (developed by analysts) as programs and test and debug them before deployment.

Advantages of Using the Database Approach

❑ Controlling redundancy in data storage and in development and maintenance efforts

- Sharing of data among multiple users.
- In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications. For example, consider the UNIVERSITY database example. Here, two groups of users might be the course registration personnel and the accounting office. In the traditional approach, each group independently keeps files on students. The accounting office keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Other groups may further duplicate some or all of the same data in their own files.
- 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. This may happen because an update is applied to some of the files but not to others. Even if an update - such as adding a new student - is applied to all the appropriate files, the data concerning the student may still be inconsistent because the updates are applied independently by each user group. For example, one user group may enter a student's birth date erroneously as 'JAN-19-1988', whereas the other user groups may enter the correct

Advantages of Using the Database Approach

- 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 space. However, in practice, it is sometimes necessary to use controlled redundancy to improve the performance of queries. For example, we may store Student_name and Course_number redundantly in a GRADE_REPORT file because whenever we retrieve a GRADE_REPORT record, we want to retrieve the student name and course number along with the grade, student number, and section identifier. By placing all the data together, we do not have to search multiple files to collect this data. This is known as **denormalization**. In such cases, the DBMS should have the capability to control this redundancy in order to prohibit inconsistencies among the files.

❑ Restricting unauthorized access to data

- When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data is often considered confidential, and only authorized persons are allowed to access such data. In addition, some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update. Hence, the type of access operation - retrieval or update - must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions. Then, the DBMS should enforce these restrictions automatically.

Advantages of Using the Database Approach

❑ Providing persistent storage for program Objects

- Object-oriented DBMSs make program objects persistent.
- 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. Programming languages typically have complex data structures, such as class definitions in C++ or Java. The values of program variables or objects are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage. When the need arises to read this data once more, the programmer must convert from the file format to the program variable or object structure. 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. Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another C++ program.

Advantages of Using the Database Approach

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. Indexes are typically based on tree data structures or hash data structures that are suitably modified for disk search. In order to process the database records needed by a particular query, those records must be copied from disk to main memory.
- 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.

Advantages of Using the Database Approach

❑ Providing backup and recovery services

- A DBMS must provide facilities for recovering from hardware or software failures. The **backup and recovery subsystem** of the DBMS is responsible for recovery.
- For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.
- Alternatively, the recovery subsystem could ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

❑ Providing multiple interfaces to different classes of users

- Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users, programming language interfaces for application programmers, etc.

Advantages of Using the Database Approach

❑ Representing complex relationships among data

- A database may include numerous varieties of data that are interrelated in many ways.
- 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 on the database

- Most database applications have certain **integrity constraints** that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints.
- The simplest type of integrity constraint involves specifying a data type for each data item.
- A more complex type of constraint that frequently occurs involves specifying that a record in one file must be related to records in other files. This is known as a **referential integrity** constraint.
- Another type of constraint specifies uniqueness on data item values, such as every student record must have a unique value for student_id. This is known as a **key** or **uniqueness** constraint.