Chapter 1: Introduction to Databases

Outline

- Introduction
  - Basic Definitions
  - Typical DBMS Functionality
  - Example of a Database
- Characteristics of the Database Approach
- Advantages of Using the Database Approach

Introduction

- **Database:** A collection of related data
  - **Data: -** Known facts

    - implicit meaning
  - **Implicit Properties**

    - represents some aspect of the real world, sometimes called the **miniworld**

    - logically coherent collection of data with some inherent meaning

    - designed, built, and populated with data for a specific purpose
  - Database

    - any size and complexity

    - may be generated and maintained manually or it may be computerized

Database Management System (DBMS):

- General-purpose software system that facilitates the processes of
  - Defining
  - constructing
  - manipulating and
  - sharing databases among various users and applications.
- Functions

  - protecting the database:

    system protection

    security protection

  - maintaining it over a long period of time

Database + DBMS software = Database System

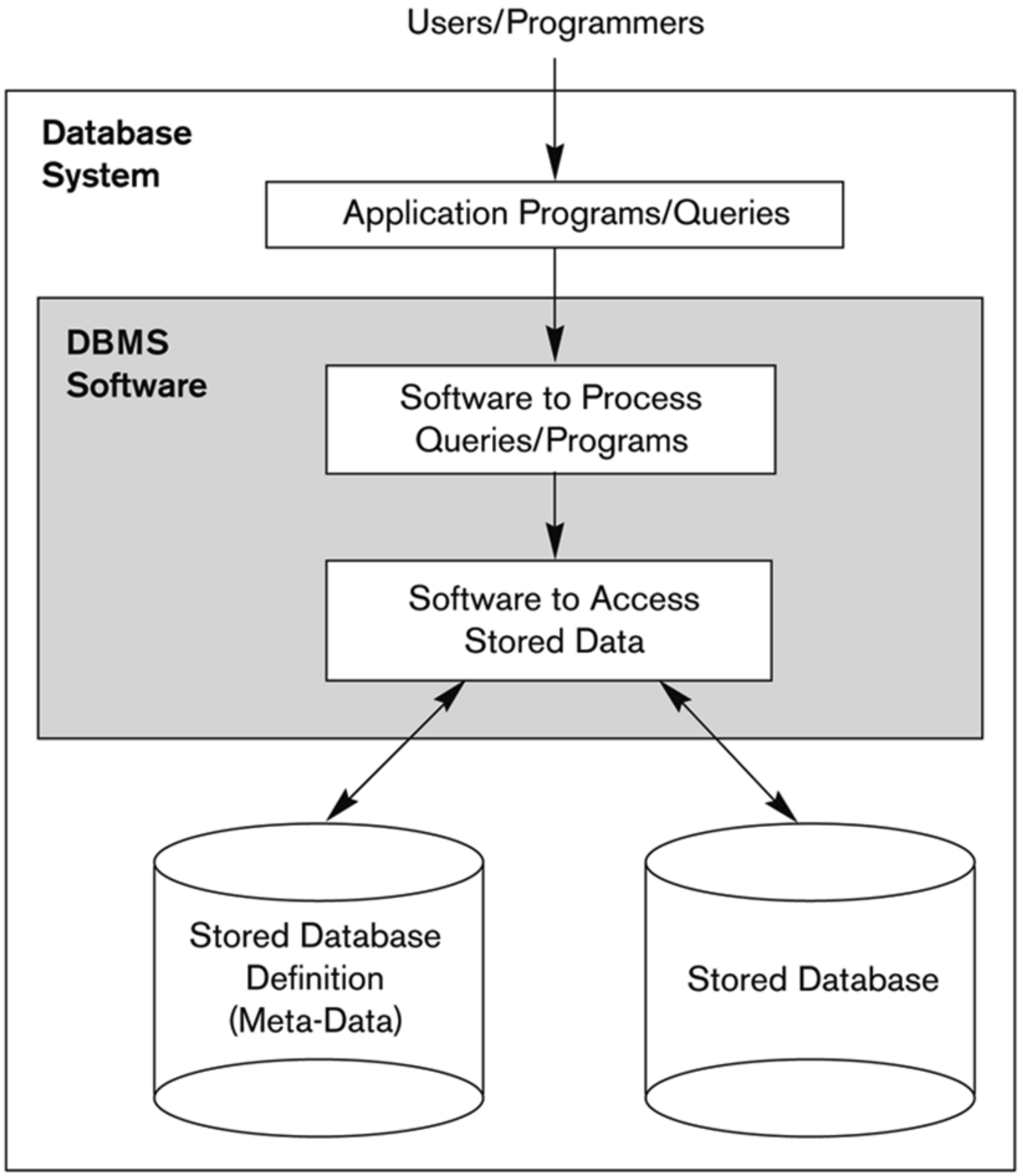

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Fig: A Simplified Database System Environment

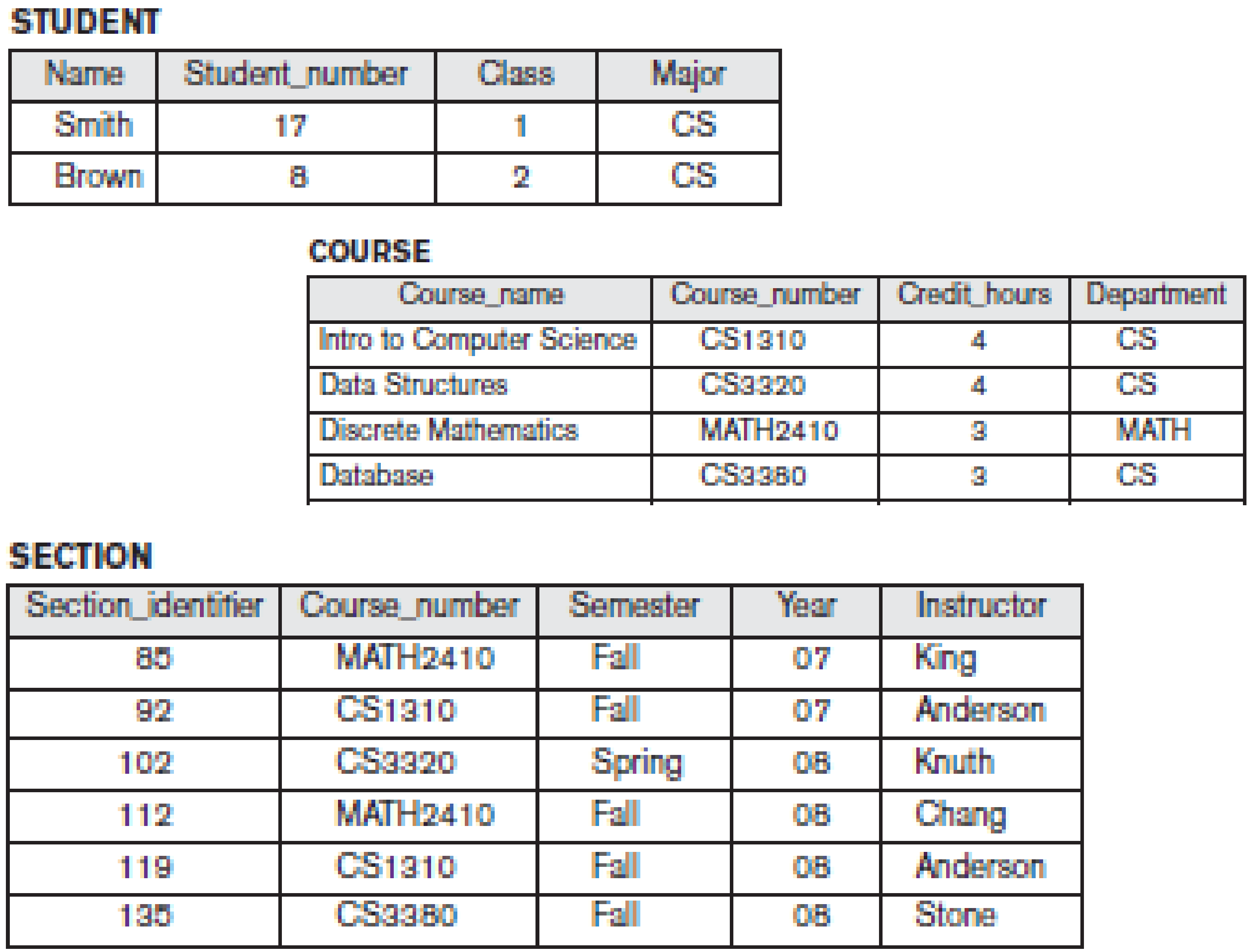
Example of a Database

- **Mini-world for the example:**
  - **UNIVERSITY** database for maintaining information concerning students, courses, and grades in a university environment
  - **Some mini-world *entities*:**

    1. STUDENT file: record on each student
    2. COURSE file: record on each course
    3. SECTION file: record on each section of a course
    4. GRADE_REPORT file: stores the grades
    5. PREREQUISITE file : stores the prerequisites of each course

Defining a UNIVERSITY database

- Specify data elements to be stored in each record
  - For example:
    - each STUDENT record includes data to represent the student’s Name, Student_number, Class, Major
    - each COURSE record includes data to represent the course_name, Course_number, Credit_hours and Department
- Specify a data type for each data element within a record
  - For example:
    - student’s Name is a string of alphabetic characters
    - student_number is an integer



STUDENT

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

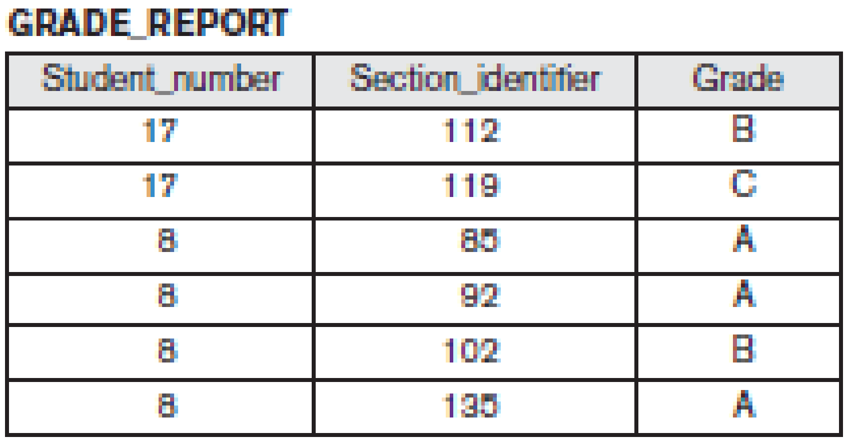
COURSE

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

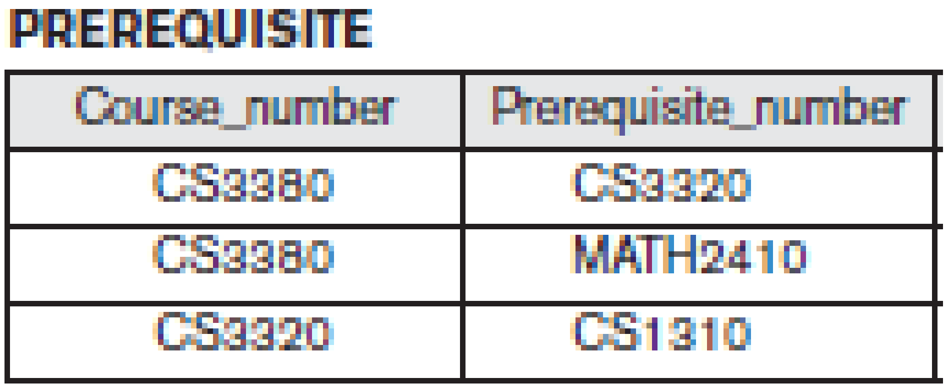
Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

Fig 1.2a : A database that stores student and course information



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

Fig 1.2b :A database that stores student and course information

Constructing the UNIVERSITY database

- store data to represent each student, course, section, grade report, and prerequisite as a record in the appropriate file
- records in the various files may be related
  - Example:
    - The record for Smith in the STUDENT file is related to two records in the GRADE_REPORT file that specify Smith’s grades in two sections.

Manipulating a UNIVERSITY database

- involves querying and updating.
  - **Examples of queries:**
    - Retrieve the transcript—a list of all courses and grades—of ‘Smith’
    - List the prerequisites of the ‘Database’ course
  - **Examples of updates :**
    - Create a new section for the ‘Database’ course for the semester fall 2008
    - Enter a grade of ‘A’ for ‘Smith’ in the ‘discrete mathematical structure’ section

Characteristics of the Database Approach

- **File Processing v/s DBMS**
  - **Redundancy**
    - 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

- 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.

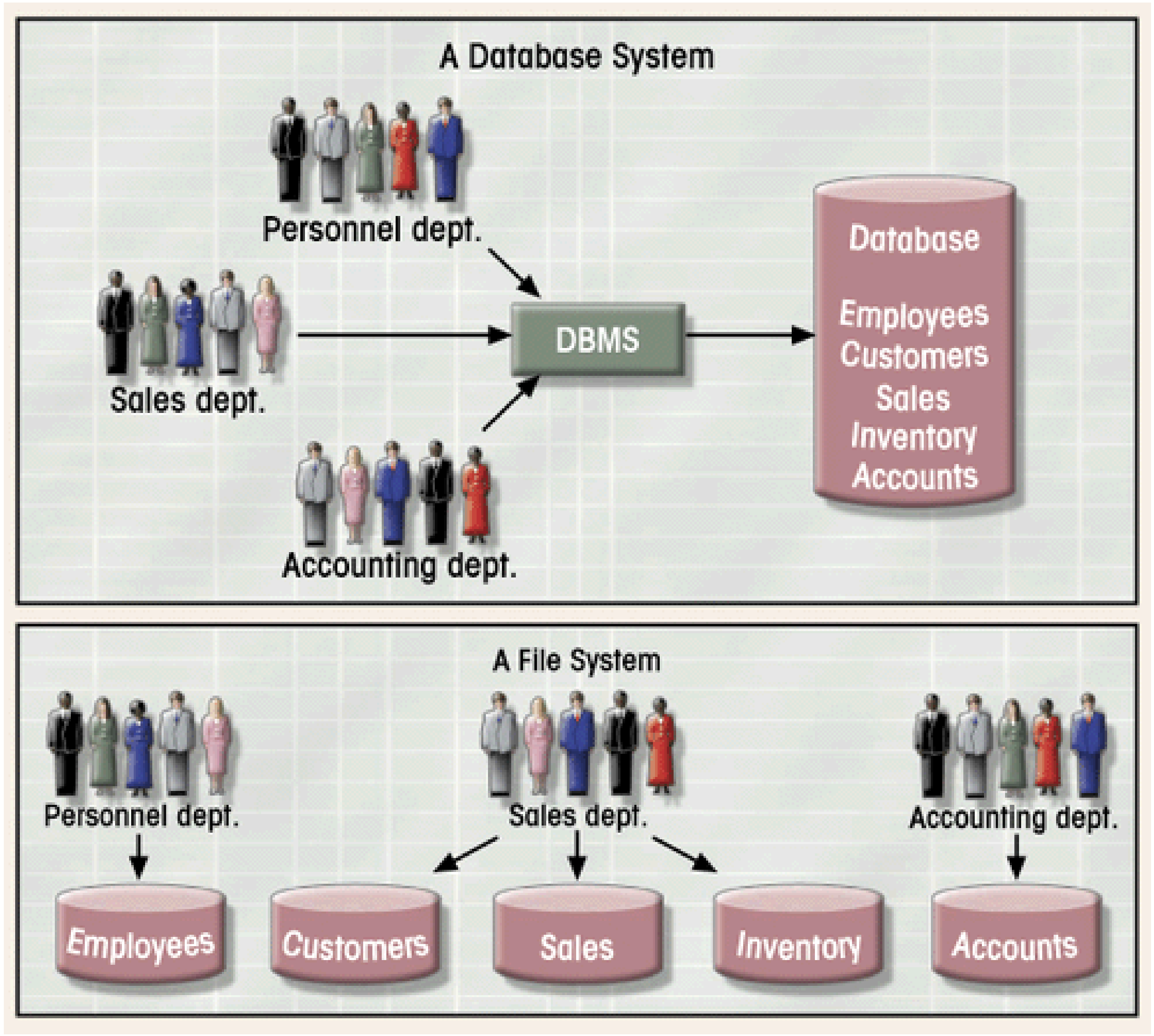
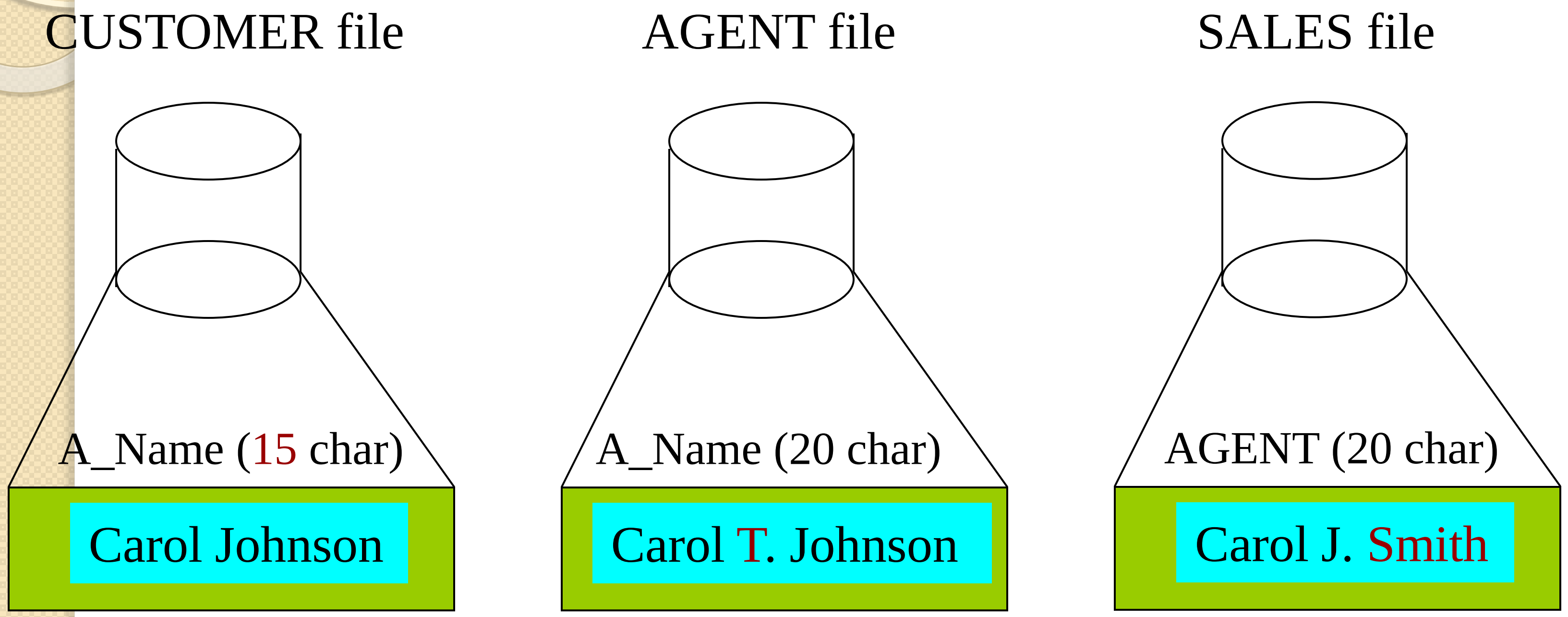


Fig 1.3: Database System vs. File System

File System: Problem Case

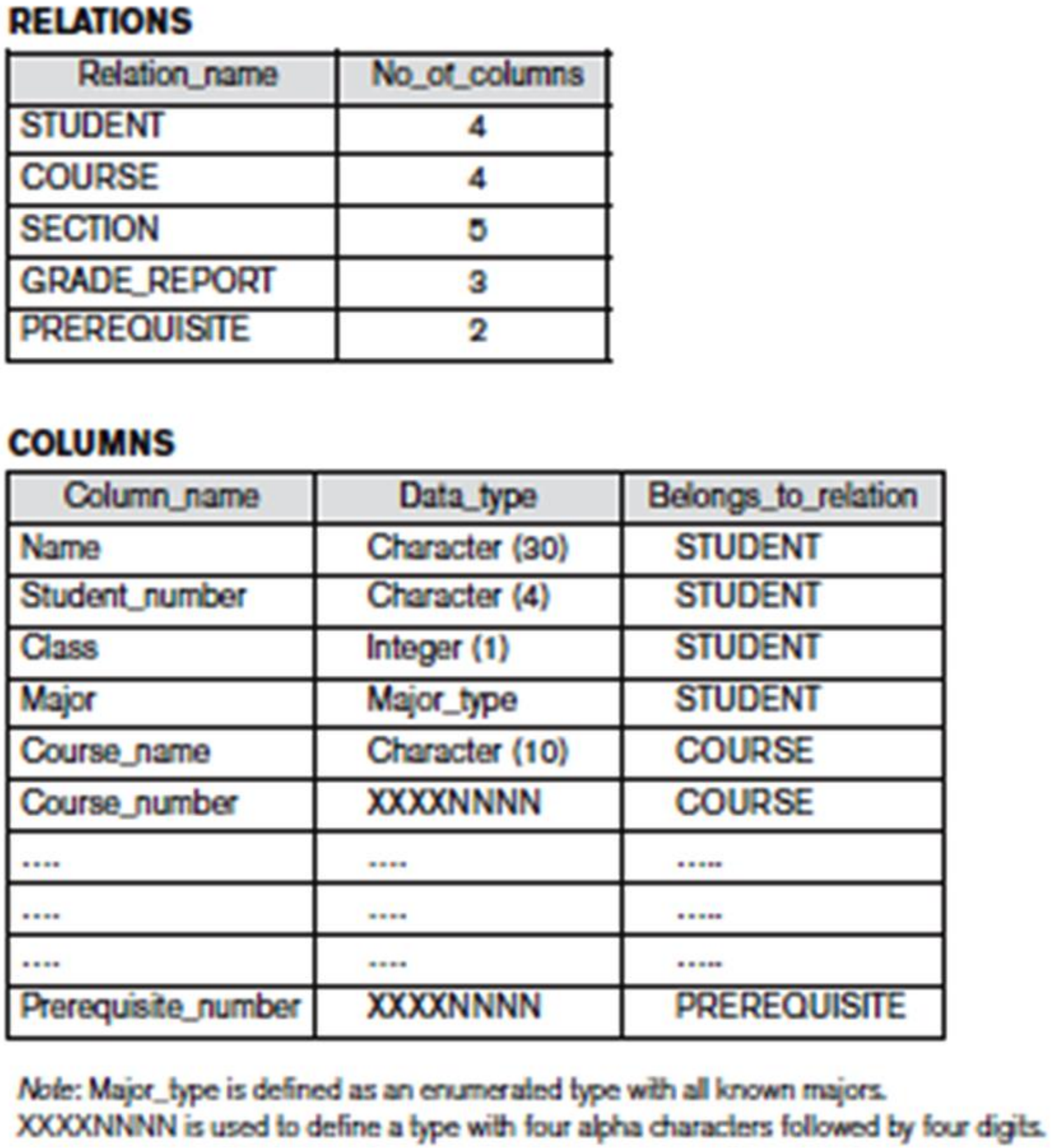


- inconsistent field name, field size
- inconsistent data values
- data duplication

- 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

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
- The information stored in the catalog is called **meta-data**
- The catalog is used by the DBMS software and also by database users who need information about the database structure
- 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.



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

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

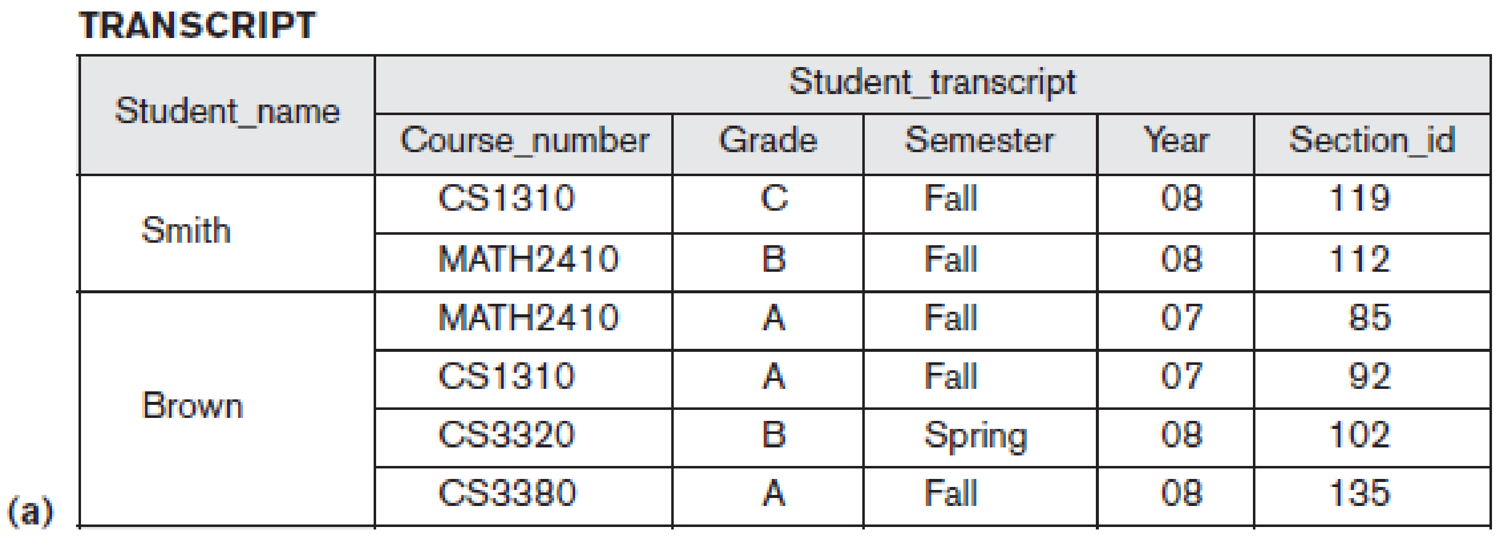
Insulation between Programs and Data, and Data Abstraction

- **Program-data independence**
  - 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.
- **Program-operation independence**
  - Users can define operations on data as part of the database definitions.
  - User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented.

- **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

Support of Multiple Views of the Data

- A database 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 that is derived from the database files but is not explicitly stored
- For example, one user of the database may be interested only in accessing and printing the transcript of each student; the view for this user is shown in Figure below



TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

Sharing of Data and Multiuser Transaction Processing

- A multiuser DBMS must allow multiple users to access the database at the same time.
- The 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.
- 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.

Advantages of Using the DBMS Approach

1. Controlling Redundancy
2. Restricting Unauthorized Access
3. Providing Persistent Storage for Program Objects
4. Providing Storage Structures and Search Techniques for Efficient Query Processing
5. Providing Backup and Recovery
6. Providing Multiple User Interfaces
7. Representing Complex Relationships among Data
8. Enforcing Integrity Constraints
9. Permitting Inferencing and Actions Using Rules
10. Additional Implications of Using the Database Approach

Advantages of Using the DBMS Approach

1. Controlling Redundancy

- Redundancy leads to several problems- duplication, wastage of storage and data may become inconsistent.
- In the database approach, the views of different user groups are integrated during database design.
- Database design that stores each logical data item in *only one place* in the database. This is known as **data normalization**, and it ensures consistency and saves storage space

- 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**.

2.Restricting Unauthorized Access

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

3. Providing Persistent Storage for Program Objects

- object-oriented database systems make it easier for complex runtime objects to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.
- Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.

4. Providing Storage Structures and Search Techniques for Efficient Query Processing

- The DBMS maintains indexes that are utilized to improve the execution time of queries and updates.
- DBMS has a **buffering** or **caching** module that maintains parts of the database in main memory buffers.
- The **query processing and optimization** module is responsible for choosing an efficient query execution plan for each query submitted to the system

5. Providing Backup and Recovery

- The **backup and recovery subsystem** of the DBMS is responsible for recovery.
- ensures that recovery is possible in the case of a system crash during execution of one or more transactions.
- Disk backup is also necessary in case of a catastrophic disk failure.

6 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.
- These include

  - query languages for casual users
  - programming language interfaces for application programmers
  - forms and command codes for parametric users
  - menu-driven interfaces and natural language interfaces for standalone users.

7. Representing Complex Relationships among Data

- A database may include numerous varieties of data that are interrelated in many ways.
- For example each section record is related to one course record and to a number of GRADE_REPORT records—one for each student who completed that section.
- 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.

8.Enforcing Integrity Constraints

- Most database applications are such that the semantics of the data require that it satisfy certain restrictions in order to make sense.
- The simplest type of integrity constraint involves specifying a data type for each data item.
  - For example, in student table we specified that the value of Name must be a string of no more than 30 alphabetic characters.
- More complex type of constraint is **referential integrity** involves specifying that a record in one file must be related to records in

  other files

  - For example, in university database, we can specify that

    every section record must be related to a course record.

- Another type of constraint specifies uniqueness on data item values, such as every course record must have a unique value for Course_number. This is known as a **key** or **uniqueness** constraint.
- It is the responsibility of the database designers to identify integrity constraints during database design.

9.Permitting Inferencing and Actions Using Rules

- In a deductive database system, one may specify declarative rules that allow the database to infer new data.
- For example figure out which students are on academic probation.
- These can be specified declaratively as rules, which when compiled and maintained by the DBMS
- In today's relational database systems, it is possible to associate triggers with tables.
- A trigger is a form of a rule activated by updates to the table
- Stored procedures
  - More involved procedures to enforce rules

10 Additional Implications of Using the Database Approach

- **Potential for Enforcing Standards**

  - permits the DBA to define and enforce standards among database users in a large organization which facilitates communication and cooperation among various departments, projects, and users within the organization.

  - Standards can be defined for names and formats of data elements, display formats, report structures and so on.

- **Reduced Application Development Time**

  - once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities

- **Flexibility**
  - It may be necessary to change the structure of a database as requirements change.
  - DBMSs allow changes to the structure of the database without affecting the stored data and the existing application programs.
- **Availability of Up-to-Date Information**
  - DBMS makes the database available to all users
  - availability of up-to-date information is essential for many transaction-processing applications, such as reservation systems or banking databases

- **Economies of Scale**

- DBMS approach permits consolidation of data and applications, to overlap between activities of data-processing in different projects or departments

- enables the whole organization to invest in more powerful processors, storage devices, or communication gear, rather than having each department purchase its equipment.

-reduces overall costs of operation and management.

Question Bank

1. Define database and briefly explain the implicit properties of the database.
2. Discuss the main Characteristics of the database approach and how does it differ from Traditional file systems?
3. Define Database Management System. Briefly discuss the advantages of using the DBMS.