Database Management Systems

許 中 川

Office: MB406, Tel: 5326

Email: hsucc@yuntech.edu.tw

Office hours: 16:00~17:30 (Tue., Fri.)

or by appointment

課程所要培養的學習目標

- 講授資料庫系統相關知識
  - 了解資料庫管理系統之架構
  - 有能力設計資訊系統所需的資料庫之架構
  - 有能力建置資料庫
  - 有能力查詢與修改資料庫內容
  - 有能力評量資料庫架構設計之優劣
  - 了解資料庫安全相關議題

此課程評量方式及其比例分配

- 三次考試（65%）
  - 兩節課考試，第三節上課(期末考週除外)
- 網路學園自我測驗（15%）
  - 每週課後網路學園自我測驗
- 小組專案作業（15%）
  - 每組四人為原則
- 課程參與（5%）
  - 課堂互動及出席率

Some Notes

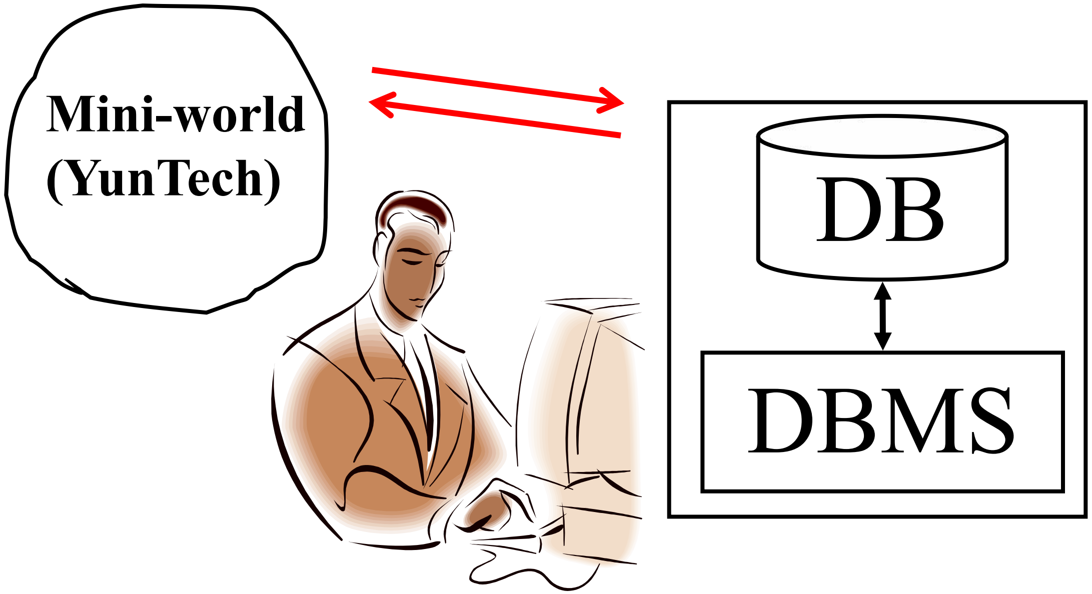
- Course materials: 課程投影片置於雲科大網路學園
- Textbook:
  - Elmasri, R. et al., *Database Systems-Models, Languages, Design, and Application Programming*, 6th ed., 2010, Pearson, 歐亞書局
- 助教: 吳俊逸，分機：5397，m10623017@yuntech.edu.tw
- Others:

1. 每周課後請至網路學園進行自我測驗。(未寫完提交前，請勿關掉測驗網頁，否則測驗結束，沒有分數，也無法重測)
2. 網路自我測驗有問題，請在測驗時間截止前向助教反應，否則無法補救
3. 考試會有英文出題，平時請務必閱讀原文課本，以免屆時看不懂題目
4. 考古題公布於網路學園，考試時考古題與網路學園題目各占約10%

Some Notes-2

1. 請把握三次測驗、線上自我測驗及作業成績
2. 無法針對私人因素，個別調整學期期末成績
3. 無法參加考試，請依學校程序請假，並事先知會老師
4. 上課前請開妥投影機及電腦，下載並開啟最新課程投影片
5. 學好本課程的要領：課前預習投影片，課後當天馬上複習上課內容
6. 上課不可睡覺、聊天、使用手機。違者每次扣兩分

What is this course all about?



1. How to place mini-world data into database?
   (1) Design the DB's structure (how and quality)
   (2) Construct the DB (define schema and load data)
2. How to retrieve data from database?
   (1) Interfaces that DBMS offers (interactive and API)

Course Content

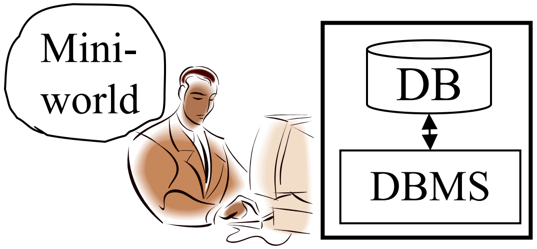
1. How to place mini-world data into database?

(1) Design the DB's structure (**how** and **quality**)

(2) Construct the DB (**define schema** and load data)

2. How to retrieve data from database?

(1) Interfaces that DBMS offers (**interactive** and **API**)



Data Model
- Ch03: The Basic (Flat) Relational Model
- Ch10: Object and Object-Relational Databases
- Ch11: XML: Concepts, Languages, and Standards

Construct Retrieve
- Ch04: SQL: Data Definition, Constraints, and Basic Queries and Updates
- Ch05: SQL: Advanced Queries, Assertions, Triggers, and Views

Design
- Ch07: Conceptual Data Modeling Using Entities
- Ch08: Mapping a Conceptual Design into a Logical Design
- Ch09: UML for Database Application Design

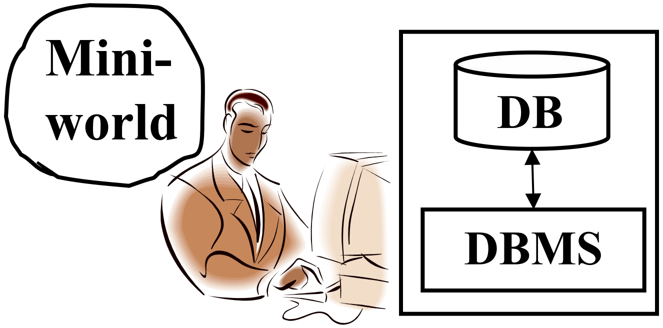
API
- Ch12: SQL Application Programming Using C and Java
- Ch13: SQL Web Programming Using C PHP

Quality
- Ch 14: Database Design Theory: Normalization and Dependencies

Chapter 1

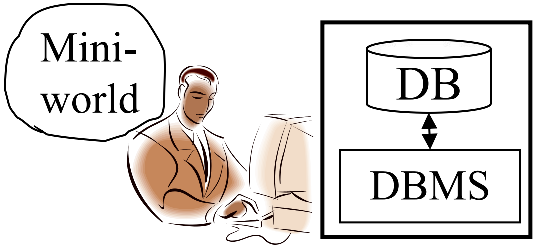
Introduction to Databases

Ch1: Introduction to Databases



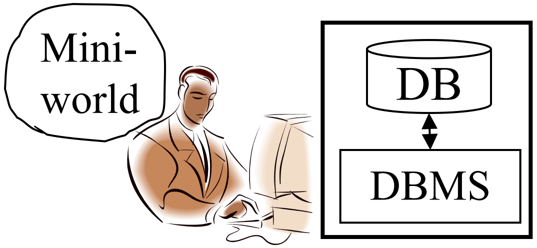
- Introduction
- An Example
- Characteristics of the DB Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of DB Applications
- When Not to Use a DBMS

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)**
  - 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.

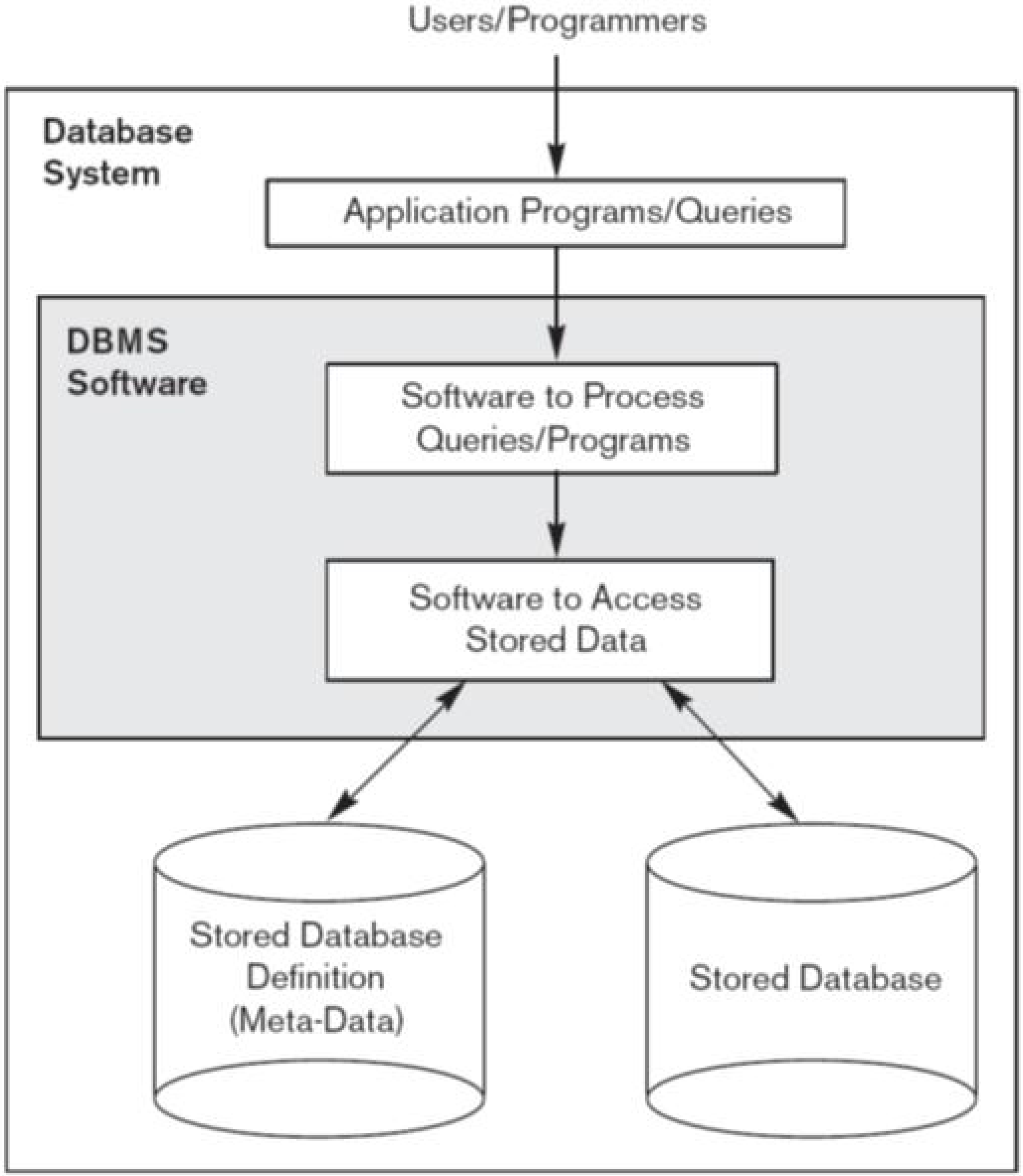
FIGURE 1.1
A simplified database system environment.



Mini-world

DB

DBMS



Users/Programmers

Database System

Application Programs/Queries

DBMS Software

Software to Process Queries/Programs

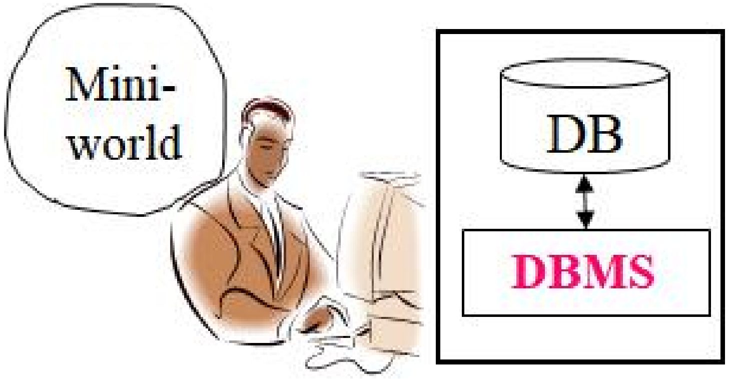
Software to Access Stored Data

Stored Database Definition (Meta-Data)

Stored Database

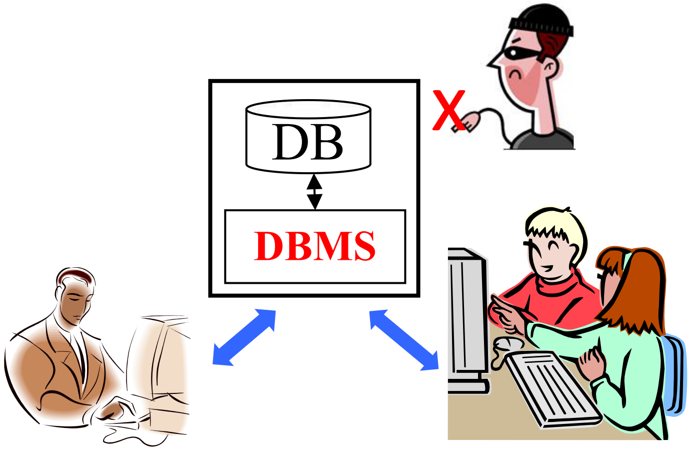
Typical DBMS Functionality-1

- Define a database
  - in terms of data types, structures and constraints約束
- Construct or Load the Database
  - on a secondary storage medium
- Manipulate(操作) the database
  - Querying查詢, generating生成, insertions插入, deletions刪除 and modifications修改 to its content內容



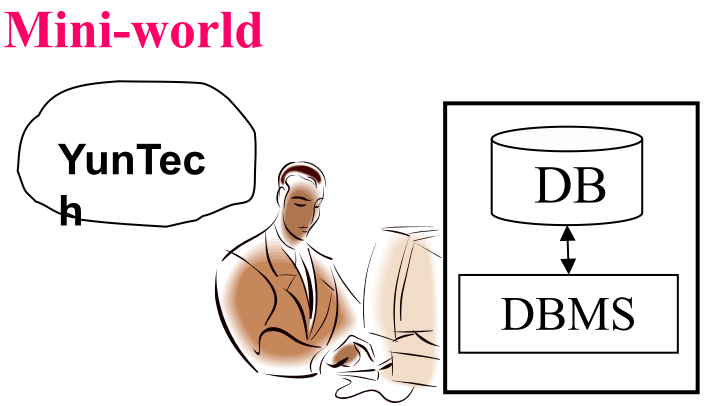
Typical DBMS Functionality-2

- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent
- Protection or Security measures to prevent unauthorized access
- “Active” processing to take internal actions on data
- Presentation and Visualization of data



Example of a Database

- **Mini-world for the example**
  - Part of a UNIVERSITY environment
  - 真實世界的資料儲存在資料庫中.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs
- **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



Mini-world

YunTech

DB

DBMS

FIGURE 1.2 A database for student and course information

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

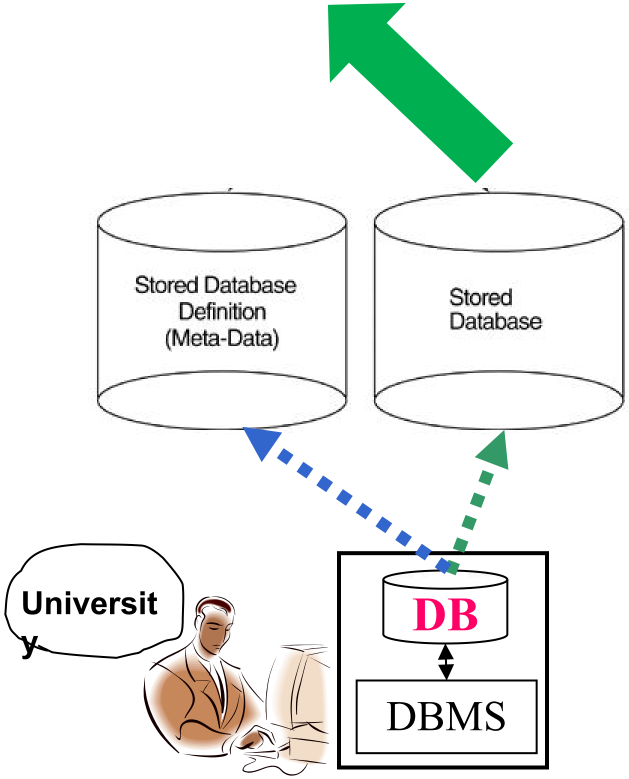
Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

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

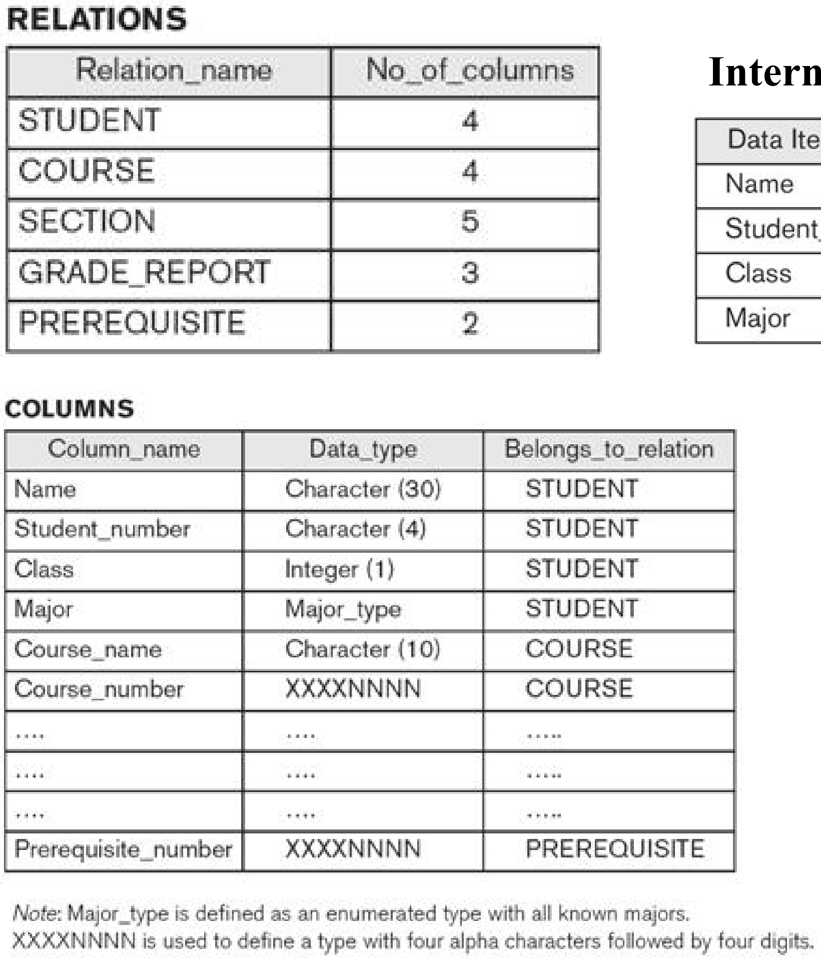
PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310



Main Characteristics of the Database Approach

- **meta-data** Self-describing
  - **meta-data** = A DBMS **catalog** stores the *description* of the database.
  - This allows the DBMS software to work with different databases.



RELATIONS

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

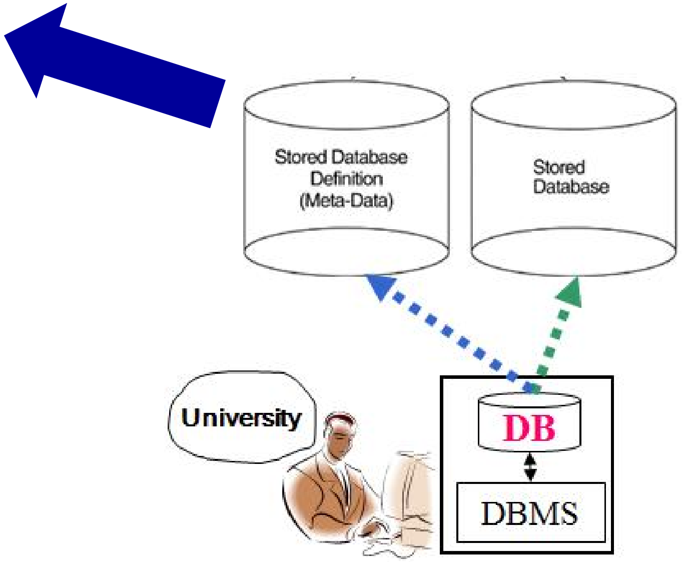
COLUMNS

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.

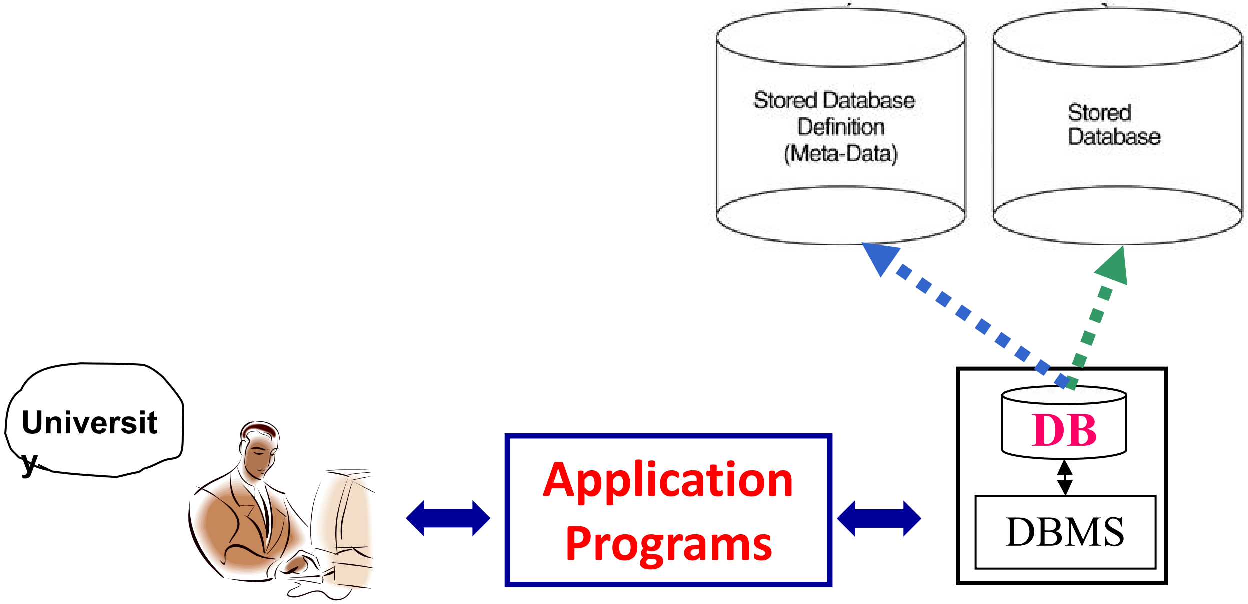
Internal storage format for a STUDENT record

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



Main Characteristics of the Database Approach

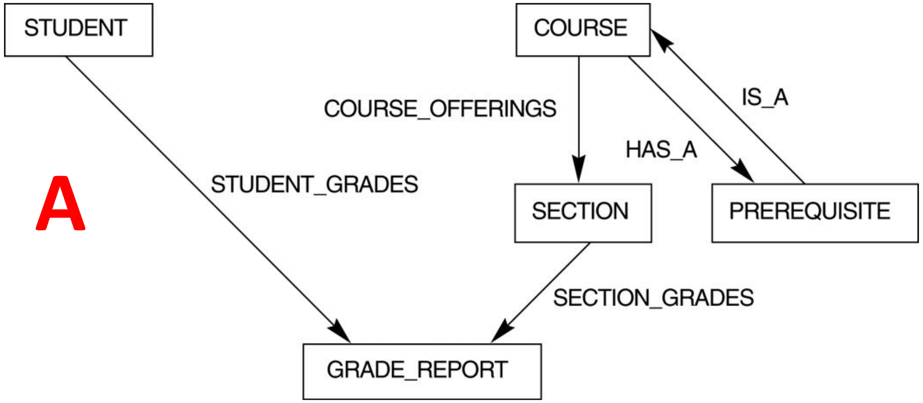
- Insulation between programs and data
  - ✓ Called **program-data independence**.
  - ✓ Allows changing "data storage structures" and "operations" without having to change the DBMS access programs訪問程序.



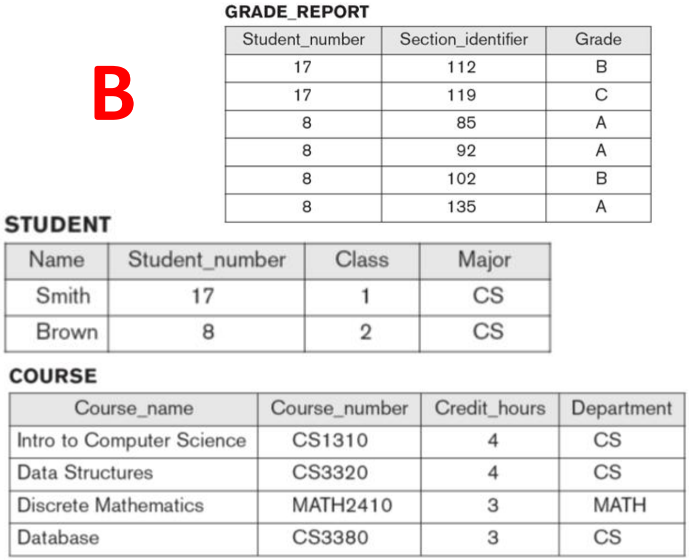
Main Characteristics of the Database Approach

- <u>Data Abstraction</u>
  - A **data model** is used to用於 hide storage details and present the users with a ***conceptual view*** of the database.

A



B



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

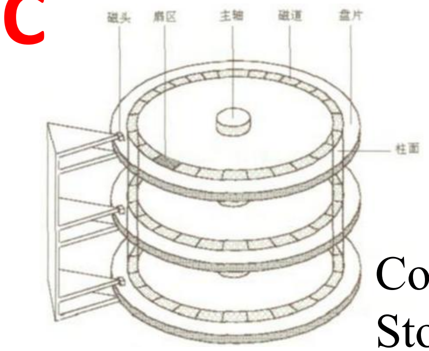
STUDENT

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

COURSE

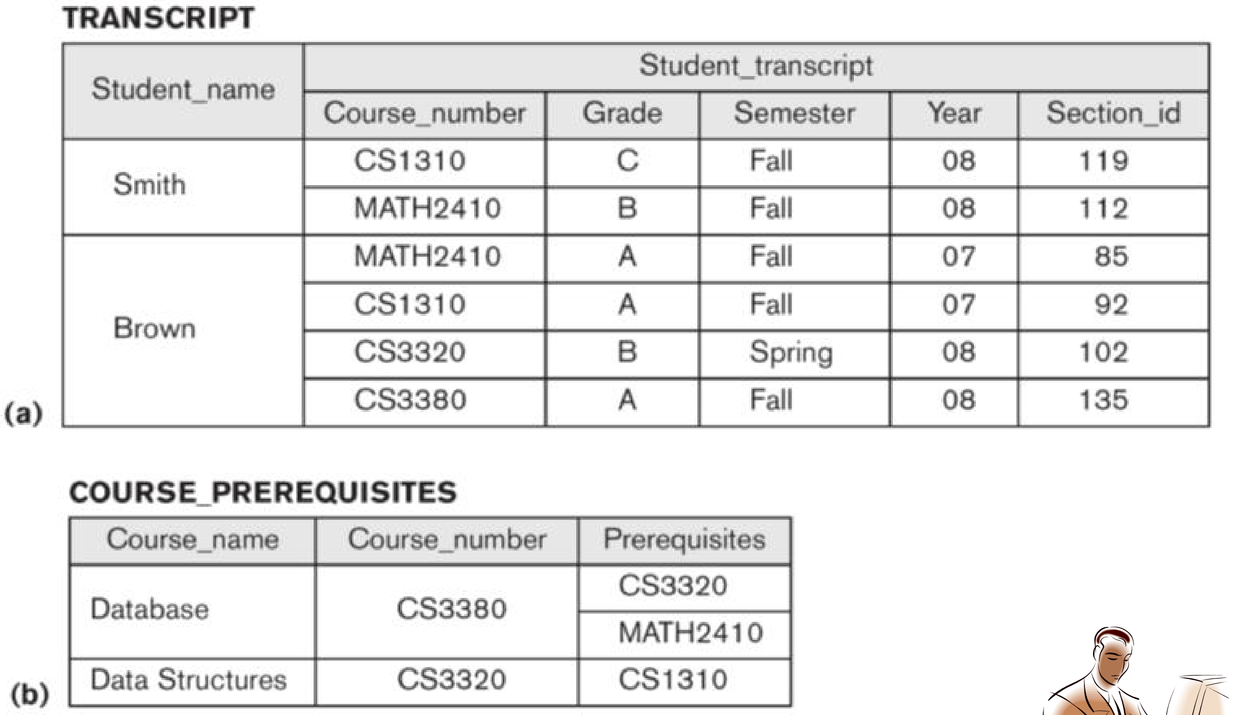
Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

C



Computer Storage

- <u>Support of multiple views of the data</u>
  – Each user may see a different view of the database, which describes *only* the data of interest to that user.



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)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

(b)

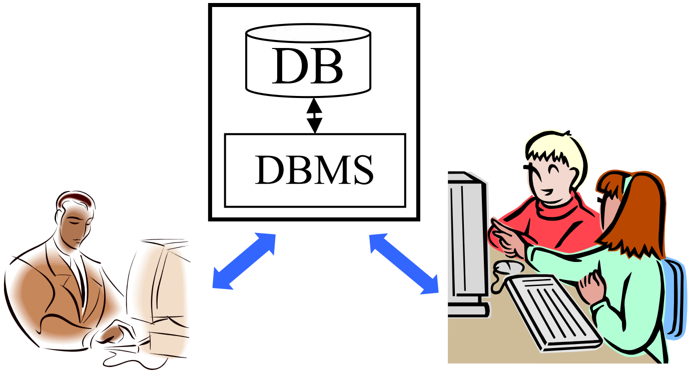
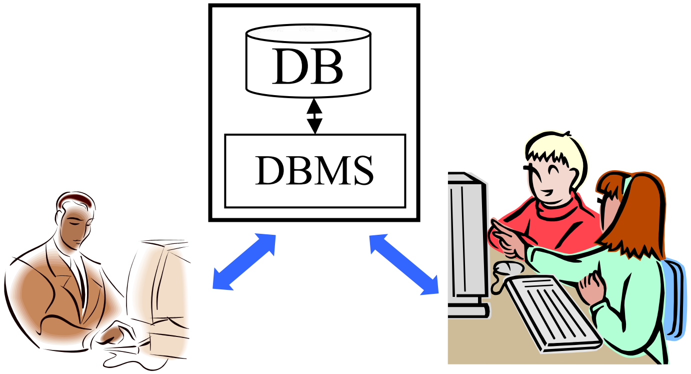


FIGURE 1.4
Two views derived from the database in Figure 1.2 (a) The STUDENT TRANSCRIPT view. (b) The COURSE PREREQUISITES view.

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.

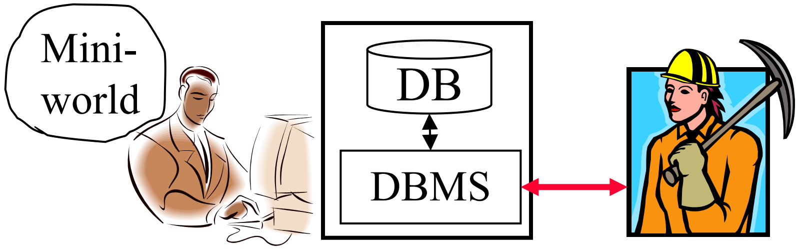


Database Users

- Users may be divided into
  - Actors on the Scene

    those who actually use and control the content
  - Workers Behind the Scene

    those who enable the database to be developed and the DBMS software to be designed and implemented.

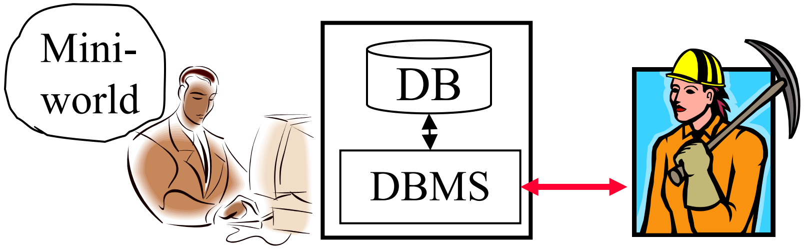


Actors on the scene

Workers behind the scene:

- DBMS system designers and implementers
- Tool developers
- Operators and maintenance personnel

Actors on the Scene



Actors on the scene

- **Database administrators:**
  - authorizing access to the database,
  - coordinating and monitoring its use,
  - acquiring software, and hardware resources, controlling its use
  - monitoring efficiency of operations.
- **Database Designers:**
  - define the content, the structure, the constraints, and functions or transactions against the database.
  - communicate with the end-users and understand their needs.
- **System Analysts and Application Programmers**
- **End-users:**

  they use the data for queries, reports and some of them actually update the database content.

Categories of End-users-1

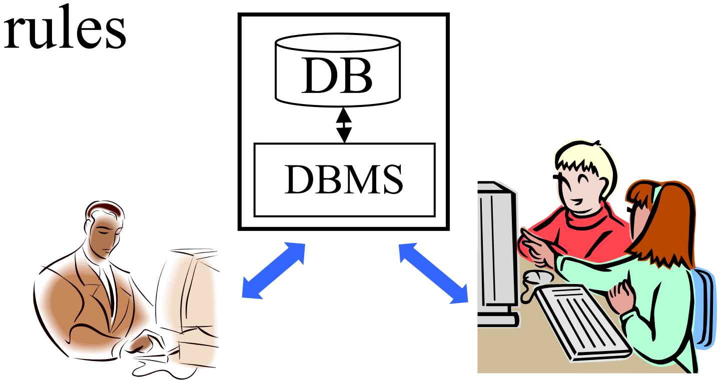
- **Casual**
  - Access database occasionally when needed; use a sophisticated DB query language;
  - Typically middle- or high-level managers
- **Naïve or Parametric**
  - They make up a large section of the end-user population. They use previously well-defined functions in the form of "canned transactions" against the database.
  - Examples are ***bank-tellers*** or ***reservation clerks*** who do this activity for an entire shift of operations.

Categories of End-users-2

- **Sophisticated**
  - *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 is *a tax program user* that creates his or her own internal database.

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
- Restricting unauthorized access to data.
- Providing persistent storage for program objects
- Providing storage structures for efficient query processing
- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Permitting inference and actions using rules

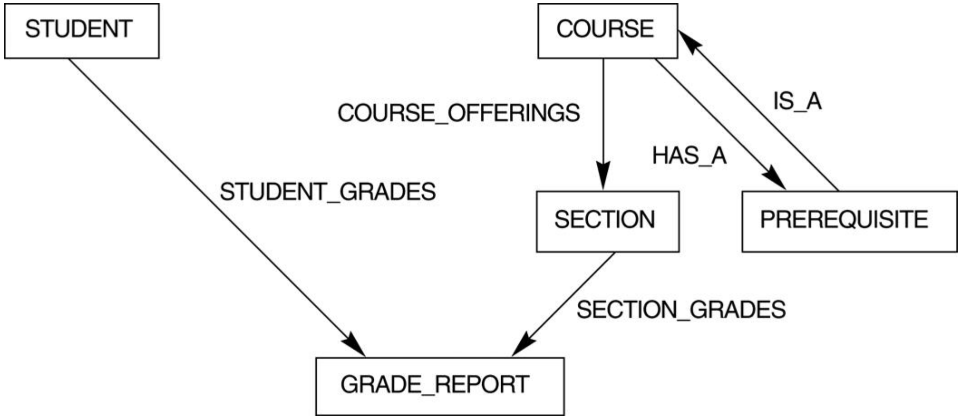


Advantages of Using the Database Approach

Permitting inference from data

ID	Name	Parent
A	Tom	John
B	John	York
C	York	Gary

Tom’s ancestors?



Representing complex relationships among data

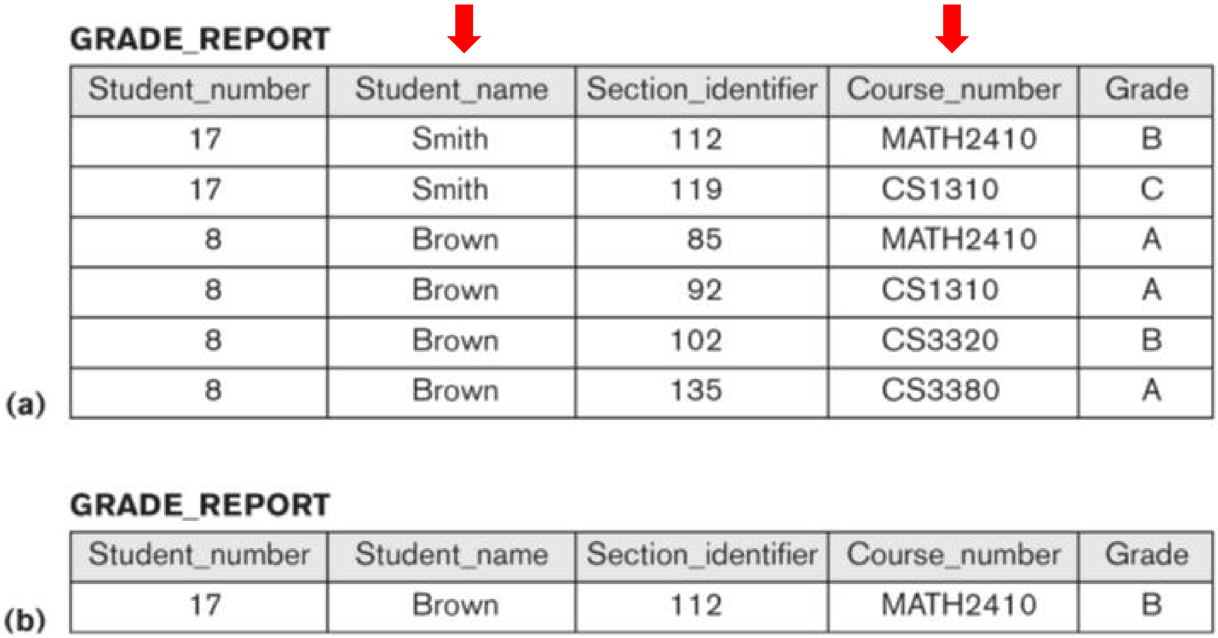
Permitting action using rules

ID	Name	MinQnt	Quantity
1	Coke	10	12
2	Pepsi	10	40
3	Latte	10	70
…	…	…	…

Trigger:
If ID.Quantity < ID.MinQnt,
then Order_Product(ID.Name)

FIGURE 1.5

Redundant storage of StudentName and CourseNumber in GRADE_REPORT. (a) Consistent data. (b) Inconsistent record.



(a)

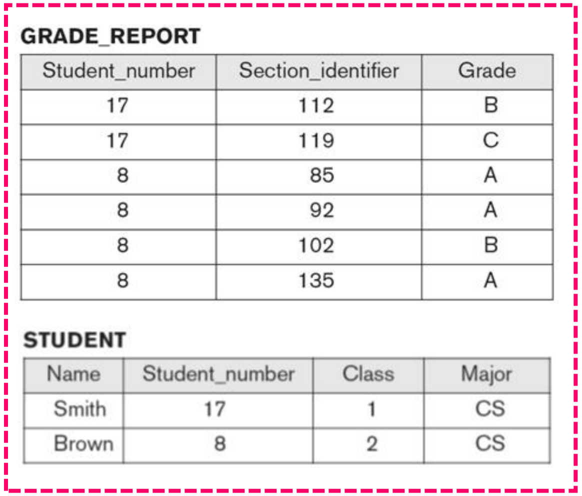
GRADE_REPORT

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

(b)

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Brown	112	MATH2410	B



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

STUDENT

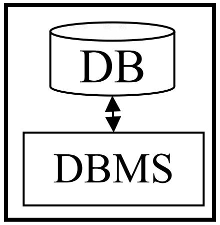
Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

DBMS can help automatically check consistency of the controlled redundancy, which is for improving query performance.

Additional Implications of Using the Database Approach

- Potential for enforcing standards
  - 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.
- 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.

- 教務處
- 學務處
- 圖書館
- …



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 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.