Course Goals

- This is the first course in database management systems.
- You will learn to:
  - Design relational databases
  - Use a relational database (SQL = Structured Query Language)
  - Build a relational database
- This course will emphasize on hands-on learning:
  - A few programming assignments: build several components of a relational database system
  - A final project: build an online database application
Prerequisite

- Data structure and algorithms
- Good at C++ programming
  - You will do considerable amount of coding in C++.
  - If you don’t know C++, you need to learn it on your own.
- English skill
  - This course is taught in English (except some TA sessions may be in Chinese).
  - If I speak too fast, please tell me to slow down.
  - You can ask questions in English or Chinese.

Administrative Details
Textbook

- The textbook is available from 新月 and (and perhaps other) bookstores.
- Other good textbooks on Database Systems:
  - “Database System Concepts” by Silberschatz, Korth, and Sudarsan

Course Format

- Lecture (2:20 ~ 3:40, 80 minutes), break (3:40 ~ 4:00, 20 minutes), and lecture (4:00 ~ 5:20, 80 minutes).
- Programming Assignments
  - SQL on Microsoft Access
  - Major database components (work in teams of 2 students)
- Exercises
  - Textbook
- Midterm and Final Exams
Grading Breakdown (Tentative)

- Tentative means that I may change it later.
- Programming Assignments & Exercises (50% of Grade)
- Midterm Exam (25% of Grade)
- Final Exam (25% of Grade)

Office Hours (Tentative) and Contact Info

- Instructor: Hao-hua Chu (朱浩華)
  - Mon 10:00~11:00, Room 527 or by appointment
  - Email: hchu@csie.ntu.edu.tw
- Teaching Assistant #1: John Huang, 劉鈺峰
  - Wed 13:00~14:00 & Thur 14:30~15:30, Lab 217
  - Email: yfhuang@csie.ntu.edu.tw
- Teaching Assistant #2, KengHao Chang, 張耿豪
  - Tues 13:00~14:00 & Fri 13:00~14:00, Room 336
  - Email: kenghao@gmail.com
Means of Communications

- Course homepage (working now)
  - Please check it daily for announcements
- BBS
  - ptt.cc, under “CSIE_DBMS” board
  - Please post your questions on BBS.
  - Please read posted messages before posting new questions.
  - Do not SPAM, you should post questions related to this course ONLY.
  - TAs will try to respond to your questions as quickly as possible.
- Send email to TAs or me.
- Come to office hours

Lecture Notes

- I will try to put lecture notes on the course homepage shortly before each lecture.
- Lecture notes are complements, not replacement of attending lecture and reading textbook.
  - Attend lectures.
  - Read assigned chapters in the textbook.
Student Evaluation of Database Course last year

- If I don’t explain materials clearly enough, please interrupt me right away!
• “Course load is too heavy”
  – This is average course load in top computer science departments.

• “Lack of midterm exam”
  – We will have a midterm exam this semester.

• “English teaching”
  – This is requested from the department & NTU university.

• “Teaching staff copying programming assignments from abroad”
  – The programming assignments come with the book.
Academic Integrity

• We detected many assignments with code plagiarism in the last year’s course.
  – Receive 0 grade for the assignment + Penalty!
• You are supposed to write your own code: every single line (except skeleton TAs give you).
• You are not allowed to
  – copy any code from other groups
  – let other groups copy your code.
  – see other groups’ code.
  – show other groups your code.
  – copy any code from the Internet & senior class (don’t even try them, because we have them!)

Any Question(s) on Administrative Things?
Talk about an interesting project in Ubiquitous Computing

(Won’t be Tested)
Chapter 1: Overview of Database Systems

Outline

• Why do we need a DBMS (Database Management System)?
• What can a DBMS do for an application?
• Why study database systems?
• Data Models: Overview of a Relational Model
• Levels of Abstraction in a DBMS
• Sample Queries in DBMS
• Transaction Management Overview
• Structure of a DBMS
Motivation for Database Systems

- Suppose that you want to build an university database. It must store the following information:
  - Students
  - Professors
  - Classes
  - Classrooms
  - Who teaches what?
  - Who teaches where?
  - Who teaches whom?

Application Requirements

- Store huge amount of data (e.g., 100+ GB) over a long period of time
- Allow people to query and update data
  - Query: what is Mary’s grade in the “Operating System” course?
  - Update: enroll Mary in the “Database” course
- Protect from unauthorized access.
  - Students cannot change their course grades.
- Protect from system crashes
  - When some system components fail (hard drive, network, etc.), database can be restored to a good state.
More Application Requirements

- Protect from incorrect inputs
  - A student cannot enroll in two courses with the same lecture times.
- Support many users to access the database at the same time
  - 100 students are using database to register courses at the same time.
- Allow administrators to easily change data schema
  - At a later time, add TA info to courses.
- Efficient database operations
  - Search for students with 5 highest GPAs

Alternative to Using a DBMS

- Store data as files in operating systems.
- Applications have to deal with the following issues:
  - 32-bit addressing (5GB) is insufficient to address 100GB+ data file
  - Write special code to support different queries
  - Write special code to protect data from multiple users and concurrent access
  - Write special code to protect against data loss / corruption as result of system crashes
  - Password-based authorization is insufficient
  - Rewrite applications when data schema changes
  - Optimize applications for efficient access and query
- Easier to buy a DBMS to handle these issues
Database Management System (DBMS)

- DBMS is software to store and manage data, so applications don’t have to worry about them.
- What can a DBMS do for applications?
  - Data Definition Language - DDL
  - Data Manipulation Language - DML
    - Query language
  - Storage management
  - Transaction Management
    - Concurrency control
    - Crash recovery
  - Provide good security, efficiency, and scalability

Why Study Database Systems?

- $$$
- They are used everywhere to store and manage data. This means $$$ for people with DB knowledge.
  - Online stores, real stores
  - Banks, credit card companies
  - Police (criminal records), Passport control
  - Airlines and hotels (reservations)
  - University, etc.
- DBMS vendors can make a lot of $$$:
  - Oracle, Microsoft (Access and SQL server), IBM (DB2), Sybase, …
Data Models

- A **data model** is a collection of concepts for describing data.
  - Entity-relation (ER) model
  - Relational model (main focus of this course)
- A **schema** is a description of a particular collection of data in a given data model.
- The **relational model** is the most widely used data model today.
  - Main concept: relation, basically a table with rows and columns of records.
  - Every relation has a schema, which describes the columns, or fields.

Relational Model

- The entire table shows an instance of the Students relation.
- The Students schema is the column heads
  - **Students(Sid: String, Name: String, Login: String, age: Integer, ...)**

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>Jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>Smith@ee</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Joe</td>
<td>Joe@cs</td>
<td>19</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Levels of Abstractions in DBMS

• Many views, one conceptual (logical) schema and one physical schema.
  – Conceptual schema defines logical structure
    • Relation tables
  – Physical schema describes the file and indexing used
    • Sorted file with B+ tree index
  – Views describe how applications (users) see the data
    • Relation tables but not store explicitly

Example: University Database

• Conceptual schema:
  – Students(sid: string, name: string, login: string, age: integer, gpa:real)
  – Courses(cid: string, cname:string, credits:integer)
  – Enrolled(sid:string, cid:string, grade:string)

• Physical schema:
  – Relations stored as unordered files.
  – Index on first column of Students.

• External Schema (View):
  – Course_info(cid:string, enrollment:integer)
Data Independence

- Three levels of abstraction provides data independence.
- This is one of the most important benefits of using DBMS.
- Applications insulated from how data is structured and stored.
  - **Logical data independence** means that when conceptual schema is changed, no need to change apps.
  - **Physical data independence** means that when physical schema is changed, also no need to change apps.

Queries in DBMS

- Sample queries on university database:
  - What is the name of the student with student ID 123456?
  - What is the average salary of professors who teach CSIE courses?
  - How many students are enrolled in database course?
- The key benefits of using a relational database are:
  - Easy to specify queries using a **query language**: Structured Query Language (SQL)
    ```sql
    SELECT S.name
    FROM Students S
    WHERE S.sid = 123456
    ```
  - Efficient query processor to get answer
Transaction Management

- A transaction is an execution of a user program in a DBMS.
- Transaction management deals with two things:
  - Concurrent execution of transactions
  - Incomplete transactions and system crashes

Concurrency Control

- Example: two travel agents (A, B) are trying to book one remaining airline seat (two transactions), only one transaction can succeed in booking.
  // num_seats is 1
  Transactions A and B: if num_seats > 0, book the seat & num_seat--; // overbook!
- Solution: use locking protocol
  Transaction A: get exclusive lock on num_seats
  Transaction B: wait until A releases lock on num_seats
  Transaction A: if num_seats > 0, book & num_seat--;
  // book the seat, num_seat is set to 0
  Transaction A: release exclusive lock on num_seats
  Transaction B: num_seats = 0, no booking; // does not book the seat
Crash Recovery

- Example: a bank transaction transfers $100 from account A to account B
  
  \[ A = A - 100 \]
  
  \(<\text{system crashes}> \quad // \text{good for the bank!}\)
  
  \[ B = B + 100 \]

- Solution: use logging, meaning that all write operations are recorded in a log on a stable storage.
  
  \[ A = A - 100 \quad // \text{recorded A value (checkpoint) in a log}\]
  
  \(<\text{system crashes}>\)
  
  \[ // \text{start recovery: read the log from disk}\]
  
  \[ A = A - 100 \]
  
  \[ B = B + 100 \]

Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and crash recovery components.
- This is one of several possible architectures; each system has its own variations.
Summary of Chapter 1

• DBMS is used to maintain and query large datasets.
• Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
• Levels of abstraction give data independence.
• A DBMS typically has a layered architecture.

Homework

• Read Chapters 1 (Today)
• Read Chapters 2 (except 2.7) & 3
Feedbacks from Students

- How much can you understand from lecture?
  - <25%
  - 50%+
  - 75%+
  - 100%

- How is the speed of my talk?
  - Too fast?
  - Too slow?
  - About right?