Why use a Relational Database?

- **Persistent data structure**
  - Large volume of data
  - "Independent" from processes using the data

- **High-level API**
  for access & modification
  - Automatically optimized

- **Transaction management (ACID)**
  - Atomicity: all or none happens, despite failures & errors
  - Consistency
  - Isolation: appearance of "one at a time"
  - Durability: recovery from failures and other errors
Updating persistent data in Java

```java
public synchronized void transferMoney(int amount, Account from, Account to) {
    try {
        from.readFromDisk();
        to.readFromDisk();
        from.decrease(amount);
        to.increase(amount);
        from.writeToDisk();
        to.writeToDisk();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Updating persistent data in SQL

```sql
BEGIN;
UPDATE accounts SET balance = balance - 100 WHERE owner = 'Mary';
UPDATE accounts SET balance = balance + 100 WHERE owner = 'John';
COMMIT;
```
Fast pace of CSE 135

- Practice, practice, practice
  - Make use of the large number of online tutorials
  - Experiment with examples to understand basic concepts

- Ask questions
  - Clarify mechanical details over email and discussion section
  - Also ask questions about concepts and big picture

- ... but not only on 9th week!
  - Many moving parts, therefore many things can go wrong.

Outline

- Database server
  - Installing Postgresql
  - Using pgAdmin

- ER Diagrams

- Practicing SQL
Installing Postgresql

- Install the Postgresql 8.4 database server
  - Choose an administrative password.
  - Ignore Stack Builder.
  - Check that your firewall blocks port 5432, so that you are the only one able to connect to the server.

Installing Postgresql

- Start the pgAdmin graphical client
  - Save the administrative password.
  - Right click on Postgresql 8.4, and select Connect
  - Right click on Databases, and select New Database
  - Enter a new name for the database, and click Okay
  - Highlight the database, and select Tools -> Query Tool
  - Write SQL code (or open the examples), and select Query -> Execute
Outline

- Database server
  - Installing Postgresql
  - Using pgAdmin

- ER Diagrams

- Practicing SQL

Example project description

- The bank has multiple branches
  - Branch: name, address, assets, ...

- The bank has many customers
  - Customer: name, address, ...

- Each customer can have arbitrarily many accounts
  - Account: account number, ...

- There are two types of accounts
  - Checking: overdraft amount, ...
  - Savings: interest rate, ...

- Each customer can apply for a loan from only one branch
  - Loan: amount, ...
Step 1: Identify entities and attributes

- **Branches**
  - Name
  - Address

- **Customers**
  - Name
  - Address

- **Loans**
  - Amount

- **Accounts**
  - Balance

Step 2: Identify relationships

- **Customers** → **Borrow** → **Loans**
  - **Deposits**
  - **Accounts**
    - Balance
  - **Loan/Branch**
    - **Account/Branch**
Step 3: Translate into SQL tables

- For every entity, create corresponding table
  - Include an ID attribute even if not in E/R
- For every relationship, create table
  - For each referenced entity $E_i$ include foreign key attribute referencing ID of $E_i$

Special case: ISA relationships
Outline

• Database server
  – Installing Postgresql
  – Using pgAdmin

• ER Diagrams

• Practicing SQL

Practicing SQL in pgAdmin

• Run and execute SQL commands
  – Add clauses incrementally
  – Maintain a history of executed commands
  – Highlight and execute

• Insert sample data

• Use aliases (even when they are optional)
**SQL aggregation problem (lecture)**

- **Problem:** List students and the number of total credits for which they have registered

```sql
SELECT s.id, s.first_name, s.last_name, SUM(e.credits)
FROM students AS s, enrollment AS e
WHERE s.id = e.student
GROUP BY s.id, s.first_name, s.last_name
ORDER BY s.id
```

**More SQL tutorials**

- Step-by-step SQL tutorials
  - http://www.firstsql.com/tutor.htm
  - http://sqlzoo.net/