Database Systems
Lecture #1 Introduction

Today's topics

- What's Database?
- Why database?
- How?

- Model and Schemas

- How to learn?

Big Names in Database Systems

Company	*Product*	*Remarks*
Oracle	Oracle 8i, 9i, etc.	World's 2nd largest software company CEO, Larry Ellison, world's 2nd richest
IBM	DB2	World's 2nd largest after Informix acquisition
Microsoft	Access, SQL Server	Access comes with MS Office
Sybase	Adaptive Server	CEO John Chen, grown up in HK
Informix	Dynamic Server	Acquired by IBM in 2001

Larry Ellison

Samuel J. Palmisano

Steve Ballmer

John Chen

What Is a Database?

- A large, integrated collection of data
- which models a real-world *enterprise:*
  - Entities
    - students, courses, instructors, TAs.
  - Relationships
    - Hillary is currently taking DB course.
    - Barack is currently teaching DB course.
    - John is currently TA-ing DB course but *took* it last year.
- A *Database Management System (DBMS)* is a software package that stores and manages DBs

Databases are everywhere: non-web

- Police station
  - Tracking crime stats.
- Airline bookings
- Retailers: Wal-Mart, etc.
  - when to re-order, purchase patterns, data-mining

Databases are everywhere: web

- Retail: Amazon, etc.
- Search engines
- Searchable DBs: IMDB, tvguide.com, etc.
- Web2.0 sites:
  - flickr = images + tags
- CMS systems (Wikis, blog & forum software, etc.)

Databases involved in ordering a pizza?

1. Pizza Hut's DB
2. Credit card records
3. CC → approval by credit agencies
4. Phone company's records
5. Caller ID
   - Error-checking, anticrime

Your wallet is full of DB *records*

- Driver's license
- Credit cards
- SYSU Card
- Medical insurance card
- Money (serial numbers)
- Etc…

"You may not be interested in databases, but databases are interested in you." - Trotsky

Example of a Traditional DB App

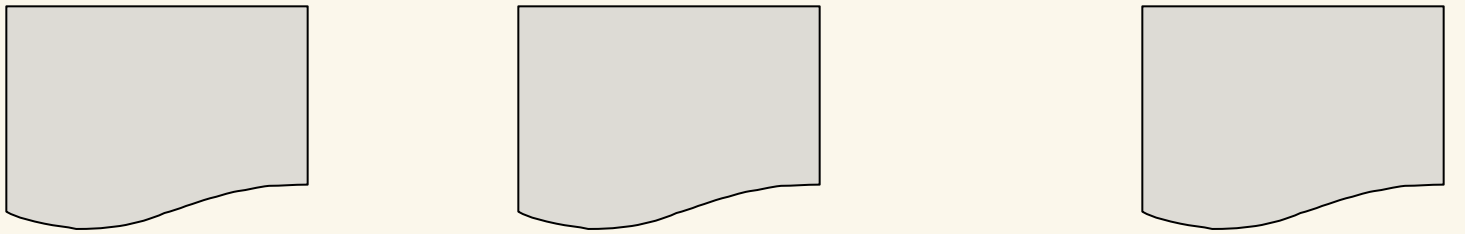
- Suppose we build a system

- We store:
  - checking accounts
  - savings accounts
  - account holders
  - state of each of each person's accounts

Can we do without a DBMS?

Sure! Start by storing the data in files:

checking.txt **savings.txt** **customers.txt**



Now write C or Java programs to implement specific tasks…

Doing it without a DBMS…

- **Transfer $100 from George's savings to checking:**

  Write a C program to do the following:

```
Read savings.txt
Find&update the line w/“George”
    balance -= 100
Write savings.txt
Read checking.txt
Find&update the line w/“George”
    balance += 100
Write checking.txt
```

Problems without an DBMS…

1. **System crashes:**

Read savings.txt
Find&update the line w/ “George.”
Write savings.txt
Read checking.txt
Find&update the line w/ “George”
Write checking.txt

CRASH!

   - **Same problem *even if reordered***
   - **High-volume → (Rare → frequent)**
2. **Simultaneous access by many users**
   - **George and Dick visit ATMs at same time**
   - **Lock checking.txt before each use–what is the problem?**

Problems without a DBMS…

3. Large data sets (100s of GBs, or TBs, …)

- No indices
  - Finding “George” in huge flatfile is expensive
- Modifications intractable without better data structures
  - “George” → “Georgie” is very expensive
  - Deletions are very expensive

Problems without an DBMS…

4. Security?

- File system may lack security features
- File system security may be coarse

5. Application programming interface (API)?

- Interfaces, interoperability

6. How to query the data?

In homebrew system, must support

- failover/recovery
- concurrent use
- deal with large datasets?
- security
- interop (互动)?
- querying in what?

- → **DBMS as application**
- Q: How does a DBMS solve these problems?
- A: See third part of course, but for now…

One big issue: Transaction processing

- Grouping of several queries (or other DB operations) into one *transaction*

- *ACID test* properties
  - Atomicity
    - all or nothing
  - Consistency
    - constraints on relationships
  - Isolation
    - concurrency control
    - *simulated solipsism (*自闭*)*
  - Durability
    - Crash recovery

Atomicity & Durability

- Avoiding inconsistent state
- A DBMS prevents this outcome
  - xacts are *all or nothing*
- One simple idea: log progress of *and plans for* each xact
  - Durability: changes stay made (with log…)
  - Atomicity: entire xact is *committed* at once

Isolation

- Many users → concurrent execution
  - Disk access is slow (compared to CPU)
  - → don't waste CPU – keep running
- Interweaving actions of different user programs
- → but can lead to inconsistency:
  - e.g., two programs simultaneously withdraw from the same account
- For each user, should look like a single-user system
  - *Simulated solipsism*

Isolation

❖ Contrast with a file in two Notepads

- Strategy: *ignore multiple users*
- whichever saves last wins
- first save is overwritten

❖ Contrast with a file in two Words

- Strategy: *blunt(*生硬的 *) isolation*
- One can edit
- To the other it's read-only

Consistency

- Each xact (on a consistent DB) must leave it in a consistent state
  - can define *integrity constraints*
  - checks that the defined claims about the data
  - Only xacts obeying them are allowed

A level up: data models (数据模型)

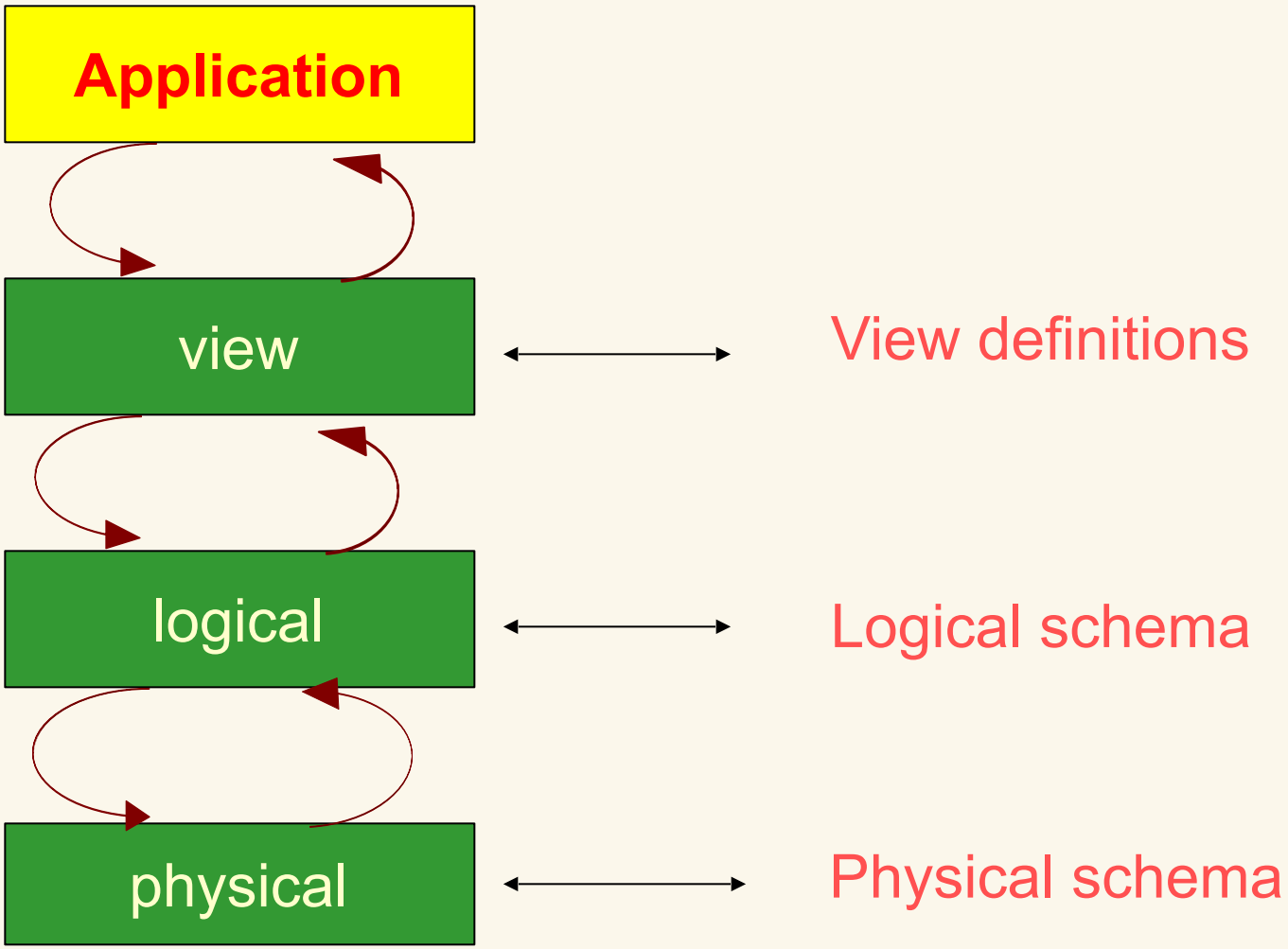
- Any DBMS uses a data model: collection of concepts for describing data
- Relational data model: basically universal
  - Oracle, DB2, SQL Server, other SQL DBMSs
    - *Relations*: table of rows & columns
  - a rel's schema (关系模式) defines its fields
- Though some have OO extensions…

Data Schemas (模式)

- Schema: description of particular set of data, using some data model

- "Physical schema"
  - Physical files on disk
- Schema
  - Set of relations/tables, with structure
- Views ("external schema")
  - Virtual tables (虚拟表) generated for user types

Level of Schemas



Schema e.g.: college registrar

- **Schema:**
  - *Students(ssn: string, name: string, login: string, age: int, gpa: real)*
  - *Courses(cid: string, cname: string, credits: int)*
  - *Enrolled(sid:string, cid:string, grade: string)*

- **Physical schema:**
  - Relations stored as unordered text files.
  - Indices on first column of each rel

- **Views:**
  - *My_courses(cname: string, grade: string, credits: int)*
  - *Course_info(ssn: string, name: string, status: string)*

How the programmer sees the DBMS

- Start with SQL DDL to *create tables*:

```
CREATE TABLE Students (
      Name CHAR(30)
      SSN CHAR(9) PRIMARY KEY NOT NULL,
      Category CHAR(20)
);
```

- Continue with SQL to *populate tables:*

```
INSERT INTO Students
VALUES('Hillary', '123456789', 'undergraduate');
```

How the programmer sees the DBMS

Students:

SSN	Name	Category
123-45-6789	Hillary	undergrad
234-56-7890	Barak	grad
	…	…

Takes:

SSN	CID	semester
123-45-6789	C20.0046	Spring, 2004
123-45-6789	C20.0056	Spring, 2004
234-56-7890	C20.0046	Fall, 2003
	…	

Courses:

CID	**CName**
C20.0046	**Databases**
C20.0056	**Advanced Software**

- **Ultimately files, but complex**

Querying: Structured Query Language

- Find all the students who have taken C20.0046:

```
SELECT SSN FROM Takes
WHERE CID='C20.0046';
```

- Find all the students who C20.0046 *previously*:

```
SELECT SSN FROM Takes
WHERE CID='C20.0046' AND
Semester='Fall, 2005';
```

- Find the students' *names*:

```
SELECT Name FROM  Students, Takes
WHERE Students.SSN=Takes.SSN AND
CID='C20.0046' AND Semester='Fall, 2005';
```

Database Industry (工业，行业，产业)

- Relational databases are based on ***set theory***
- Commercial DBMSs: Oracle, IBM's DB2, Microsoft's SQL Server, etc.
- Opensource: MySQL, PostgreSQL, etc.
- DBAs manage these
- Programmers write apps (**CRUD**, etc.)
- XML ("semi-structured data") also important

The Study of DBMS

- Primary aspects:
  - Data modeling
  - SQL
  - DB programming
  - DBMS implementation
- This course covers all four (tho less of #4)
- Also will look at some more advanced areas
  - XML, websearch, column-oriented DBs, RAID, RegExs, MapReduce…

Course outline

- **Database design:**
  - Entity/Relationship models
  - Modeling constraints
- **The relational model:**
  - Relational algebra
  - Transforming E/R models to relational schemas
- **SQL**
  - DDL & query language

Course outline

- Programming for databases
- Some DB implementation
  - Indexes, sorting, xacts
- Advanced topics…

- May change as course progresses
  - partly in response to audience

- Also “current events”
  - Slashdot/whatever, Database Blog, etc.

Summary - what is this course about?

- Languages: SQL (some XML …)
- Data modeling
- Some theory! (rereading…)
  - Functional dependencies, normal forms
  - e.g., how to find most efficient schema for data
- Some DB*MS* implementation (algs & data structs)
- Algorithms and data structures (in the latter part)
  - e.g., indices make data much faster to find – how?
- Lots of DB implementation and hacking for the project

For next time

- Get the book
- Skim chapter 1
- Start reading chapter 2

Review Questions

- What's Database?
- What problems a DBMS can solve?
- What's transaction?
- What's the ACID properties of transaction
- What's data model?
- What's data schemas?