

## Introduction

CPS 196.3  
Introduction to Database Systems

## Course goals

- ❖ Random things you might do (for fun or profit) after taking this course
  - Develop your own database-driven Web sites (like Amazon, eBay, etc.)
  - Write a toy version of Google
  - Use a database system for many things you used to do in Excel, Outlook, etc.
  - Explain to friends why MySQL, despite its coolness, is not a “real” database system
  - Upgrade your Web sites with XML
  - ... ..

## Course roadmap

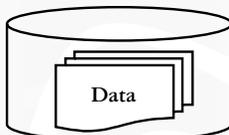
- ❖ Relational databases
  - Relational algebra, database design, SQL, application programming
- ❖ Data warehousing and data mining
- ❖ XML
  - Data model and query languages, application programming, interplay between XML and relational databases
- ❖ Database internals
  - Storage, indexing, query processing and optimization, concurrency control and recovery
- ❖ Web data management
  - Web crawling, keyword searches, page ranking

## What is a database system?

From Oxford Dictionary:

- ❖ Database: an organized body of related information
- ❖ Database system, DataBase Management System (DBMS): a software system that facilitates the creation and maintenance and use of an electronic database

## What do you want from a DBMS?



- ❖ Answer queries (questions) about data
- ❖ Update data
- ❖ And keep data around (persistent)

## An example

- ❖ Bank database: Each account has a number, an owner, a balance, ...
- ❖ Query: What's the balance in Homer Simpson's account?
- ❖ Update: Homer withdraws \$100
- ❖ Persistency: Homer will be pretty upset if his balance disappears after a power outage

## Sounds simple!

7

```
... ..
00987-00654#Ned Flanders#2500.00
00123-00456#Homer Simpson#400.00
00142-00857#Montgomery Burns#1000000000.00
... ..
```

- ❖ ASCII file
- ❖ Accounts separated by newlines
- ❖ Fields separated by #'s

## Query

8

```
... ..
00987-00654#Ned Flanders#2500.00
00123-00456#Homer Simpson#400.00
00142-00857#Montgomery Burns#1000000000.00
... ..
```

- ❖ What's the balance in Homer Simpson's account?
- ❖ A simple script
  - Scan through the file
  - Look for the line containing "Homer Simpson"
  - Print out the balance

## Performance problems

9

- ❖ Tens of thousands of accounts are not Homer's
- ☞ Cluster accounts: Those owned by "A..." go into file A; those owned by "B..." go into file B; etc.
  - Change the script to decide which file to search
- ☞ Keep the accounts sorted by owner name
  - Change the script to do binary search
- ☞ Index accounts by owner name: Index entries have the form  $\langle \text{owner\_name}, \text{file\_offset} \rangle$ 
  - Change the script to search the index for file offset
- ❖ What happens when the query changes to: Which accounts have 0 balance?

## Observations

10

- ❖ Many ways to boost performance by changing the organization of data
- ❖ Different ways make sense for different scenarios
- ❖ What is wrong?
  - Access to data is not declarative
  - Whenever data is reorganized, we need to reprogram all applications

## Physical data independence

11

- ❖ Applications should not need to worry about how data is physically structured and stored
- ❖ Applications should work with a logical data model and declarative query language
- ❖ Leave the implementation details and optimization to DBMS
- ❖ The single most important reason behind the success of DBMS today
  - And a Turing Award for E. F. Codd

## Solution

12

- ❖ Relational data model
  - Data is stored in relations (tables)
  - ☞ Digression: What's a data model?
    - Describes conceptual structuring of data
    - Another example of a data model is XML: Data is stored as an XML document, with tagged, nested "elements"
- ❖ Relational query languages: relational algebra, relational calculus, SQL, Datalog, etc.
  - Support declarative operations on relations (selection, join, etc.)

## Another example

13

- ❖ Account: number, owner, balance, branch\_id, ...
- ❖ Branch: branch\_id, location, ...
- ❖ Query: Who have accounts with 0 balance managed by a branch in Springfield?

## Before relational “revolution”

14

- ❖ Simplified (!) CODASYL (circa 1960's)

```
Account.balance := 0
FIND Account RECORD BY CALC-KEY
FIND OWNER OF CURRENT Account-Branch SET
IF Branch.location = "Springfield" THEN
  PRINT Account.owner
```

- Assume that we can quickly find accounts by balance
- Assume there is a link from accounts to branches

- ❖ Programmer controls “navigation,” but the best navigation method depends on availability of indexes, actual data distribution, etc.
  - How about navigating from branches to accounts?
- ☞ Does not provide physical data independence

## After relational “revolution”

15

- ❖ SQL (1970's – present)

```
SELECT Account.owner
FROM Account, Branch
WHERE Account.balance = 0
AND Branch.location = 'Springfield'
AND Account.branch_id = Branch.branch_id;
```

- ❖ Programmer specifies what answers a query should return, but not how the query is executed
- ❖ DBMS picks the best execution strategy based on availability of indexes, actual distribution of the data, etc.
- ☞ Provides physical data independence

## Major DBMS today

16

- ❖ Oracle
- ❖ IBM DB2 (from System R, System R\*, Starburst)
- ❖ Microsoft SQL Server
- ❖ NCR Teradata
- ❖ Sybase
- ❖ Informix (acquired by IBM)
- ❖ PostgreSQL (from UC Berkeley's Ingres, Postgres)
- ❖ Tandem NonStop (acquired by Compaq, now HP)
- ? MySQL and Microsoft Access



## Modern DBMS features

17

- ❖ Persistent storage of data
- ❖ Logical data model; declarative queries and updates
  - physical data independence
    - Relational model is the dominating technology today
    - XML is a hot wanna-be
- ☞ What else?

## DBMS is multi-user

18

- ❖ Example

```
get account balance from database;
if balance > amount of withdrawal then
  balance = balance - amount of withdrawal;
dispense cash;
store new balance into database;
```
- ❖ Homer at ATM1 withdraws \$100
- ❖ Marge at ATM2 withdraws \$50
- ❖ Initial balance = \$400, final balance = ?
  - Should be \$250 no matter who goes first

## Final balance = \$300

19

Homer withdraws \$100:

```
read balance; $400
```

```
if balance > amount then
  balance = balance - amount; $300
write balance; $300
```

Marge withdraws \$50:

```
read balance; $400
if balance > amount then
  balance = balance - amount; $350
write balance; $350
```

## Final balance = \$350

20

Homer withdraws \$100:

```
read balance; $400
```

```
if balance > amount then
  balance = balance - amount; $300
write balance; $300
```

Marge withdraws \$50:

```
read balance; $400
```

```
if balance > amount then
  balance = balance - amount; $350
write balance; $350
```

## Concurrency control in DBMS

21

- ❖ Appears similar to concurrent programming problems?
  - But data not main-memory variables
- ❖ Appears similar to file system concurrent access?
  - Approach taken by MySQL  
(fun reading: <http://openacs.org/philosophy/why-not-mysql.html>)
  - But want to control at much finer granularity
    - Or else one withdrawal would lock up all accounts!

## Recovery in DBMS

22

- ❖ Example: balance transfer
  - decrement the balance of account X by \$100;
  - increment the balance of account Y by \$100;
- ❖ Scenario 1: Power goes out after the first instruction
- ❖ Scenario 2: DBMS buffers and updates data in memory (for efficiency); before they are written back to disk, power goes out
- ❖ Log updates; undo/redo during recovery

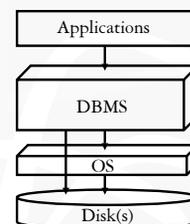
## Summary of modern DBMS features

23

- ❖ Persistent storage of data
- ❖ Logical data model; declarative queries and updates  
→ physical data independence
- ❖ Multi-user concurrent access
- ❖ Safety from system failures
- ❖ Performance, performance, performance
  - Massive amounts of data (terabytes ~ petabytes)
  - High throughput (thousands ~ millions transactions per minute)
  - High availability ( $\geq 99.999\%$  uptime)

## Modern DBMS architecture

24



- ❖ Many details will be filled in the DBMS box

## People working with databases

25

- ❖ End users: query/update databases through application user interfaces (e.g., Amazon.com, 1-800-DISCOVER, etc.)
- ❖ Database designers: design database “schema” to model aspects of the real world
- ❖ Database application developers: build applications that interface with databases
- ❖ Database administrators (a.k.a. DBA’s): load, back up, and restore data, fine-tune databases for performance
- ❖ DBMS implementors: develop the DBMS or specialized data management software, implement new techniques for query processing and optimization inside DBMS

## Course information

26

- ❖ Book
  - *Database Systems: The Complete Book*, by H. Garcia-Molina, J. D. Ullman, and J. Widom.
- ❖ Web site
  - <http://www.cs.duke.edu/courses/fa1102/cps196.3/>
  - Course information
  - Syllabus and reading assignments
  - Lecture slides, assignments, programming notes
- ❖ Blackboard for emails, discussion, grades
  - Please make sure you can get emails through Blackboard

## Course load

27

- ❖ 4 homework assignments (30%)
- ❖ Programming project (30%)
  - Details to be given in the third week of class
- ❖ Midterm (20%)
- ❖ Final (20%)
  - Comprehensive, but with emphasis on the second half of the course

## Survey

28

Please respond “yes” or “no” to the following:

1. Have you ever programmed in C++ or Java?
2. Have you ever worked with a DBMS? (Microsoft Access and MySQL also count.)
3. Have you ever written a multi-table SQL query?
4. Have you ever written XQuery?
5. Are you familiar with a B-tree?
6. Hash join?
7. Outerjoin?
8. Do you know why a DBMS cares about inferring  $A = C$  from  $A = B$  and  $B = C$ ?