Introduction

CompSci 316
Introduction to Database Systems

A few words about myself (and databases)

- ❖ Have been doing (and enjoying) research in databases ever since grad school (1995)
  - Didn't take any database course as an undergrad
- ☞ Now, why would you want to take 316?
- ☞ It's not really about databases per se—it's about principles of data management
- ❖ E.g., Google might not care if you know SQL, but…
  - They still ask you "big data" questions in interviews
  - Brin was a grad student in the Stanford Database Group

Trend: Moore's Law reversed

- ❖ Moore's Law: *Processing power doubles every 18 months*
- ❖ Amount of data doubles every 9 months
  - Disk sales (# of bits) doubles every 9 months
  - Parkinson's Law: *Data expands to fill the space available for storage*
  - As of 2009, Facebook ingests 15 terabytes of data per day and maintains a 2.5-petabyte data warehouse
  - CERN's Large Hadron Collider will produce 15 petabytes per year
- ☞ Moore's Law reversed:
- ❖ Does your attention span double every 18 months?
  - No, so we need smarter data management techniques

Democratizing data (and analysis)

4

- And it's not just about money and science
- Democratization of data: more data—relevant to you and the society—are becoming available
  - "Government in the sunshine": spending reports, school performance, crime reports, corporate fillings, campaign contributions, …
  - "Smart planet": sensors for phones and cars, roads and bridges, buildings and forests, …
- But few people know how to analyze them
- You will learn how to help bridge this divide

Misc. course information

5

- Website: `http://www.cs.duke.edu/courses/fall12/compsci316/`
  - Course information; tentative syllabus and reference sections in the book; lecture slides, assignments, programming notes
- Book: *Database Systems: The Complete Book*, by H. Garcia-Molina, J. D. Ullman, and J. Widom. 2nd Ed.
- Gradiance: see course website for sign-up information
- Sakai: for grades only
- Mailing list: `compsci316@cs.duke.edu`
  - Messages of general interest only
- No "official" recitation sessions; help sessions for assignments, project, and exams to be scheduled
- TA: Rozemary Scarlat

Grading

6

[90%, 100%]	A- / A / A+
[80%, 90%)	B- / B / B+
[70%, 80%)	C- / C / C+
[60%, 70%)	D
[0%, 60%)	F

- No "curves"
- Scale may be adjusted downwards (i.e., grades upwards) if, for example, an exam is too difficult
- Scale will not go upwards—mistake would be mine alone if I made an exam too easy

Course load

7

- Four homework assignments (35%)
  - Including Gradiance as well as additional written and programming problems
- Course project (25%)
  - Details to be given in the third week of class
- Midterm and final (20% each)
  - Open book, open notes
  - Final is comprehensive, but emphasizes the second half of the course

Example past projects from last year

8

- Chumchi: a social website with *relevant* feeds
  - Kirill Klimuk (also PickyU)
- FriendsTracker app: where are my friends?
  - Anthony Lin, Jimmy Mu, Austin Benesh, Nic Dinkins
- LocalBug: marketplace for local farmers
  - Ashley Chou, Ross Cahoon
- FoodTr@cker: where is that yummy food truck?
  - Rohan Kshirsagar, Brandon Millman, Faith Xu
- MovieShare: who borrowed my DVD?
  - Glenn Rivkees

More past examples

9

- ePrint iPhone app
  - Ben Getson and Lucas Best, 2009
- Making iTunes social
  - Nick Patrick, 2006; Peter Williams and Nikhil Arun, 2009
- Duke Schedulator: ditch ACES—plan schedules visually!
  - Alex Beutel, 2008
- SensorDB: manage/cleanse/visualize sensor data from Duke Forest
  - Ashley DeMass, Jonathan Jou, Jonathan Odom, 2007
- Facebook$^+$
  - Tyler Brock and Beth Trushkowsky, 2005
- Web-based K-ville tenting management
  - Zach Marshall, 2005

So, what is a database system?

10

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?

11

- Keep data around (persistent)
- Answer questions (queries) about data
- Update data

- Example: a traditional banking application
  - Data: Each account belongs to a branch, has a number, an owner, a balance, …; each branch has a location, a manager, …
  - Persistency: Balance can't disappear after a power outage
  - Query: What's the balance in Homer Simpson's account? What's the difference in average balance between Springfield and Capitol City accounts?
  - Modification: Homer withdraws $100; charge accounts with lower than $500 balance a $5 fee

Sounds simple!

12

```
1001#Springfield#Mr. Morgan
```

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

- ASCII file
- Accounts/branches separated by newlines
- Fields separated by #'s

Query

13

```
1001#Springfield#Mr. Morgan
... ...
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 accounts file
  - Look for the line containing "Homer Simpson"
  - Print out the balance

Query processing tricks

14

- Tens of thousands of accounts are not Homer's
- What happens when the query changes to: What's the balance in account 00142-00857?

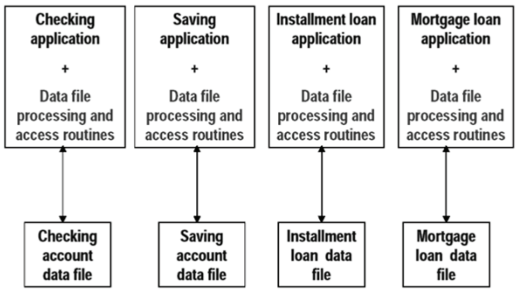
Observations

15

- Tons of tricks (not only in storage and query processing, but also in concurrency control, recovery, etc.)
- Different tricks may work better in different usage scenarios (example?)
- Same tricks get used over and over again in different applications

16

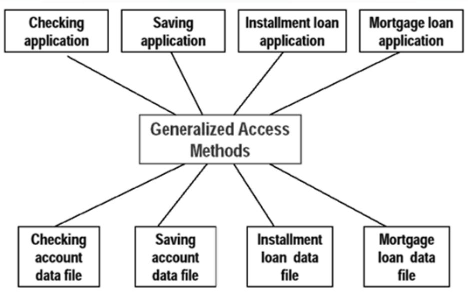
The birth of DBMS – 1



(Figure from Hans-J. Schek's *VLDB* 2000 slides)

17

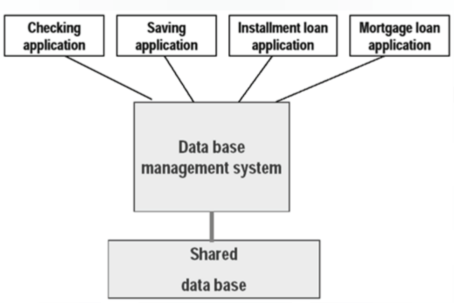
The birth of DBMS – 2



(Figure from Hans-J. Schek's *VLDB* 2000 slides)

18

The birth of DBMS – 3



(Figure from Hans-J. Schek's *VLDB* 2000 slides)

Early efforts

19

- "Factoring out" data management functionalities from applications and standardizing these functionalities is an important first step
  - CODASYL standard (circa 1960's)
  - ☞Bachman got a Turing award for this in 1973
- But getting the abstraction right (the API between applications and the DBMS) is still tricky

CODASYL

20

- Query: Who have accounts with 0 balance managed by a branch in Springfield?
- Pseudo-code of a CODASYL application:

```
Use index on account(balance) to get accounts with 0 balance;
For each account record:
  Get the branch id of this account;
  Use index on branch(id) to get the branch record;
  If the branch record's location field reads "Springfield":
    Output the owner field of the account record.
```

- Programmer controls "navigation": accounts → branches
  - How about branches → accounts?

What's wrong?

21

- The best navigation strategy & the best way of organizing the data depend on data/workload characteristics
- With the CODASYL approach
  - To write correct code, programmers need to know how data is organized physically (e.g., which indexes exist)
  - To write efficient code, programmers also need to worry about data/workload characteristics
  - ☞Can't cope with changes in data/workload characteristics

The relational revolution (1970's)

22

- A simple data model: data is stored in relations (tables)
- A declarative query language: SQL

```
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, data/workload characteristics, etc.

☞ Provides physical data independence

Physical data independence

23

- 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 in 1981

Standard DBMS features

24

- Persistent storage of data
- Logical data model; declarative queries and updates $\rightarrow$ physical data independence
  - Relational model is the dominating technology today
  - XML has been a hot wanna-be

☞ What else?

DBMS is multi-user

25

- 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

26

Homer withdraws $100: | Marge withdraws $50:

```
read balance; $400
                                 read balance; $400
                                 if balance > amount then
                                     balance = balance - amount; $350
                                     write balance; $350
if balance > amount then
    balance = balance - amount; $300
    write balance; $300
```

Final balance = $350

27

Homer withdraws $100: | Marge withdraws $50:

```
read balance; $400
                                 read balance; $400
if balance > amount then
    balance = balance - amount; $300
    write balance; $300
                                 if balance > amount then
                                     balance = balance - amount; $350
                                     write balance; $350
```

Concurrency control in DBMS

28

- Similar to concurrent programming problems?
  - But data not main-memory variables
- Similar to file system concurrent access?
  - Approach taken by MySQL in the old days (fun reading: `http://openacs.org/philosophy/why-not-mysql.html`)
  - Still used by SQLite (as of Version 3)

Recovery in DBMS

29

- 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
- How can DBMS deal with these failures?

Summary of standard DBMS features

30

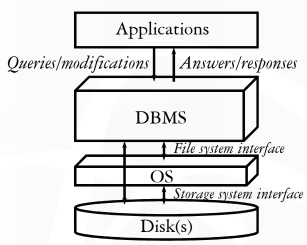
- Persistent storage of data
- Logical data model; declarative queries and updates $\rightarrow$ physical data independence
- Multi-user concurrent access
- Safety from system failures
- Performance, performance, performance
  - Massive amounts of data (terabytes $\sim$ petabytes)
  - High throughput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

Major DBMS today

- Oracle
- IBM DB2 (from System R, System R*, Starburst)
- Microsoft SQL Server
- Teradata
- Sybase (acquired by SAP)
- Informix (acquired by IBM)
- PostgreSQL (from UC Berkeley's Ingres, Postgres)
- Tandem NonStop (acquired by Compaq, now HP)
- MySQL (acquired by Sun, then Oracle)

? SQLite
? Microsoft Access
? BerkeleyDB (acquired by Oracle)

DBMS architecture today

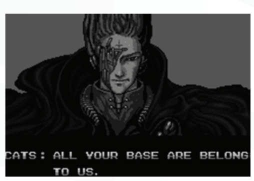


- Much of the OS is bypassed for performance and safety
- We will be filling in many details for the DBMS box

AYBABTU?

"Us" = relational databases

- Most data is not in them!
  - Personal data, web, scientific data, system data, …
- "NoSQL" movement
  - Less structure, less consistency
  - More flexibility, more availability, more scalability



(Use of AYBABTU inspired by Garcia-Molina)

- This course will look beyond relational databases

34

Course components

- Relational databases
  - Relational algebra, database design, SQL, app programming
- XML
  - Data model and query languages, app programming, interplay between XML and relational databases
- Database internals
  - Storage, indexing, query processing and optimization, concurrency control and recovery
- Topics beyond databases (TBD)
  - Privacy in data publishing, data warehousing and data mining, Web search, indexing, MapReduce, etc.