Introduction

CPS 116

Introduction to Database Systems

2

A few words about myself (and databases)

- Have been doing (and enjoying) research in databases ever since grad school (1995)
- But didn't take any database course as an undergrad
  - Just didn't appreciate it
- ☞ Now, why would you want to take 116?

3

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*
  - E.g., Facebook ingests 15 terabytes of data per day and maintains a 2.5-petabyte data warehouse!
- ☞
- Does your attention span double every 18 months?
  - No, so we need smarter data management techniques

Misc. course information

4

- Course website: `http://www.cs.duke.edu/courses/fall09/cps116/`
  - 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 ("Online Accelerated Learning"): see course website for purchase information
- Blackboard: for grades only
- Mailing list: `cps116@cs.duke.edu`
  - Messages of general interest only
- No "official" recitation sessions; help sessions for assignments, project, and exams to be scheduled
- Say hi to our TA, Dongtao Liu

Grading

5

[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

6

- 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

7

- Duke Schedulator: ditch ACES—plan your schedule visually!
  - Alex Beutel, 2008
- SensorDB: managing, cleansing, and visualizing sensor data collected from the Duke Forest
  - Ashley DeMass, Jonathan Jou, Jonathan Odom, 2007
- SuperDatabase: GUI for creating schema with rich datatypes, as well as editing and querying such data
  - Andy Ewing, MacRae Linton, Congyi Wu, and David Zhang, 2007
- yourTunes: social music networking
  - Nick Patrick, 2006
- Facebook$^+$
  - Tyler Brock and Beth Trushkowsky, 2005
- Web-based K-ville tenting management
  - Zach Marshall, 2005

A few projects ideas for this semester

8

- Computational journalism
  - Media's watchdog role is at risk because of traditional media's decline ⇒ leveraging computer science to help saving investigative journalism
- ERS: making it easy for non-programmers to model and manage semi-structured data
  - Duke immunologists are interested in using this system to track their computational and experimental workflows
- … and more (see me during office hours)

So, what is a database system?

9

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

10

What do you want from a DBMS?

- Keep data around (persistent)
- Answer queries (questions) 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 account with lower than $500 balance with a $5 fee

11

Sounds simple!

```
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

12

Query

```
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

13

- What happens when the query changes to: What's the balance in account 00142-00857?

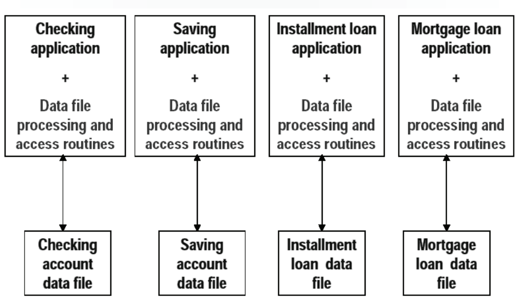
Observations

14

- 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

The birth of DBMS – 1

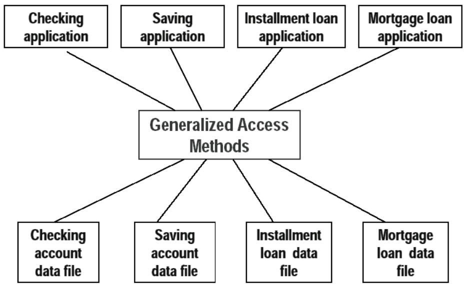
15



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

16

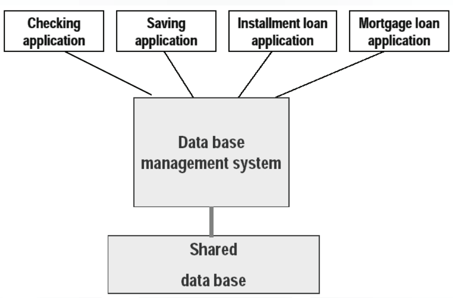
The birth of DBMS – 2



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

17

The birth of DBMS – 3



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

18

Early efforts

- "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

19

- 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?

20

- The best navigation strategy & the best way of organizing the data depend on data/workload characteristics
- With the CODASYL approach

The relational revolution (1970’s)

21

- 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

22

- 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

Modern DBMS features

23

- 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

24

- 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

25

Homer withdraws $100:

```
read balance; $400
```

Marge withdraws $50:

```
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

26

Homer withdraws $100:

```
read balance; $400
```

Marge withdraws $50:

```
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

27

- Appears similar to concurrent programming problems?
  - But data not main-memory variables
- Appears 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)

28

Recovery in DBMS

- 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?

29

Summary of modern DBMS features

- 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 ~ petabytes)
  - High throughput (thousands ~ millions transactions per minute)
  - High availability ($\geq$ 99.999% uptime)

30

Major DBMS today

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

? Microsoft Access

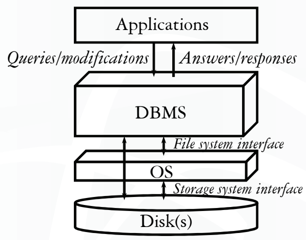
? SQLite

? BerkeleyDB (acquired by Oracle)

Relational inside

Modern DBMS architecture

31



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

AYBABTU?

32

("us" = relational databases)

- ❖ Most of the data is not in relational databases!
  - Personal data
  - Web
  - Scientific data
  - System data
- ❖ Data management is expanding to these areas
  - This course will look beyond relational databases too

(Use of AYBABTU inspired by Garcia-Molina)

Course components

33

- ❖ 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
  - Privacy in data publishing
  - Data warehousing and data mining
  - Web search and indexing
  - etc.