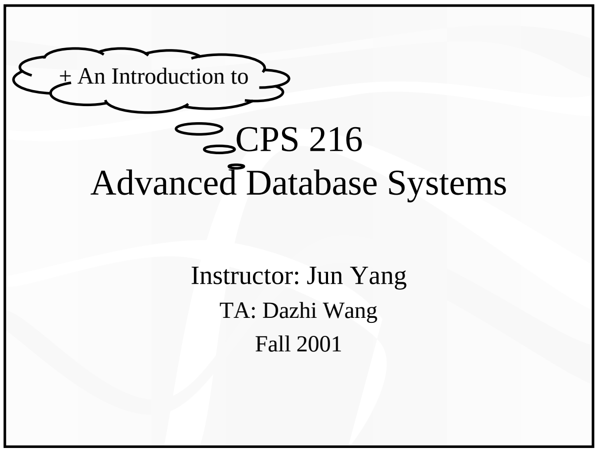


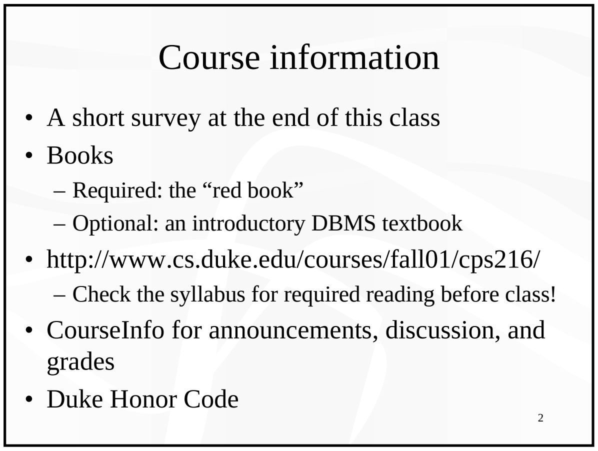
± An Introduction to

CPS 216
Advanced Database Systems

Instructor: Jun Yang

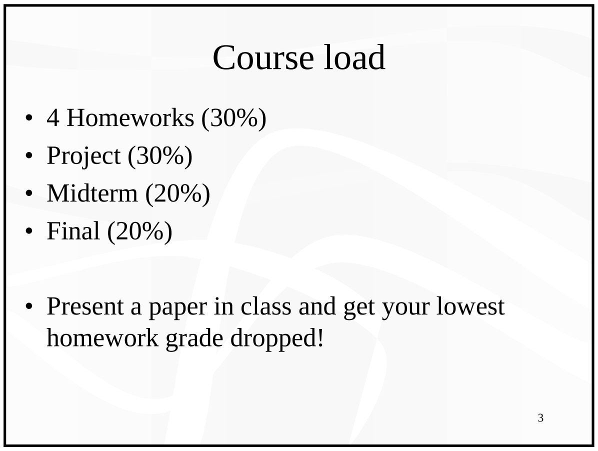
TA: Dazhi Wang

Fall 2001



Course information

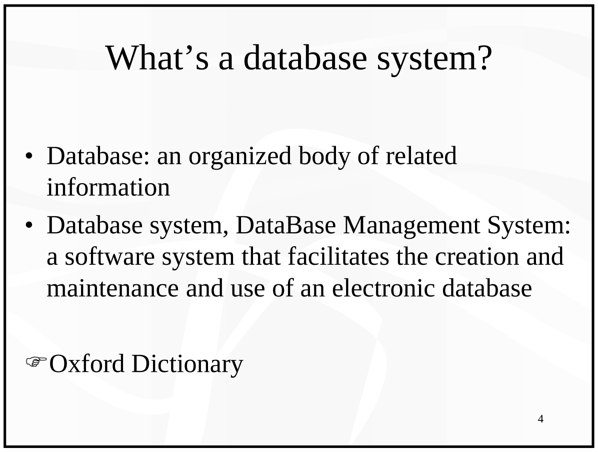
- A short survey at the end of this class
- Books
  - Required: the “red book”
  - Optional: an introductory DBMS textbook
- http://www.cs.duke.edu/courses/fall01/cps216/
  - Check the syllabus for required reading before class!
- CourseInfo for announcements, discussion, and grades
- Duke Honor Code



Course load

- 4 Homeworks (30%)
- Project (30%)
- Midterm (20%)
- Final (20%)

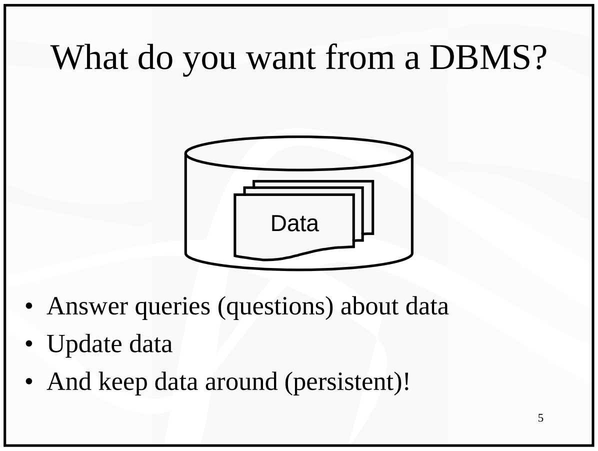
- Present a paper in class and get your lowest homework grade dropped!



What’s a database system?

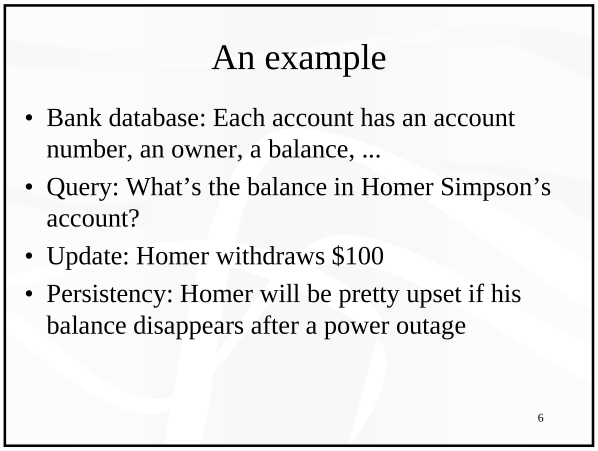
- Database: an organized body of related information
- Database system, DataBase Management System: a software system that facilitates the creation and maintenance and use of an electronic database

☞Oxford Dictionary



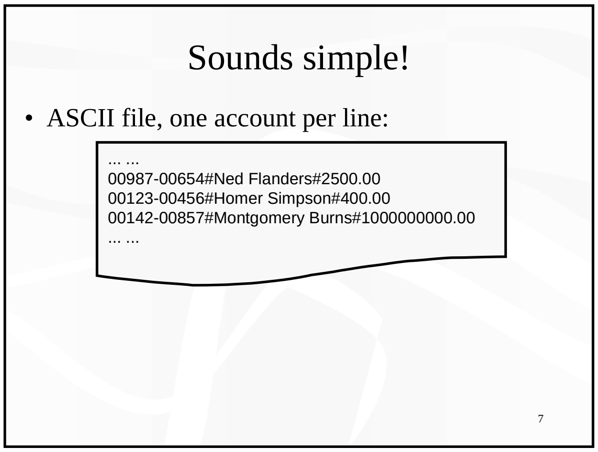
What do you want from a DBMS?

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



An example

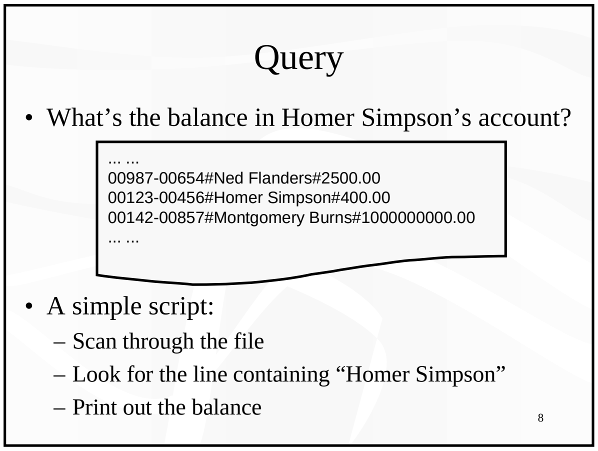
- Bank database: Each account has an account number, an owner, a balance, ...
- Query: What’s the balance in Homer Simpson’s account?
- Update: Homer withdraws $100
- Persistency: Homer will be pretty upset if his balance disappears after a power outage



Sounds simple!

- ASCII file, one account per line:

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

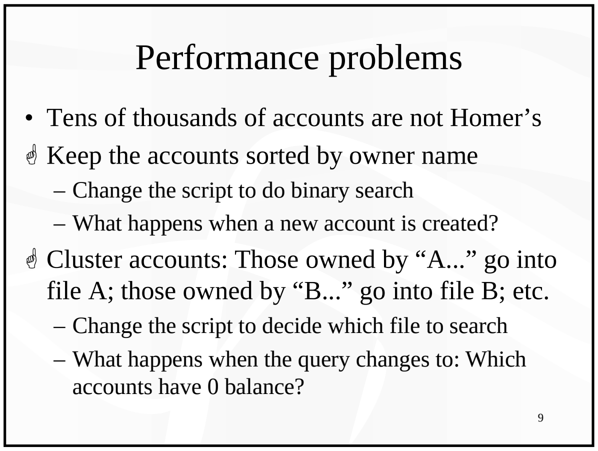


Query

- What’s the balance in Homer Simpson’s account?

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

- A simple script:
  - Scan through the file
  - Look for the line containing “Homer Simpson”
  - Print out the balance

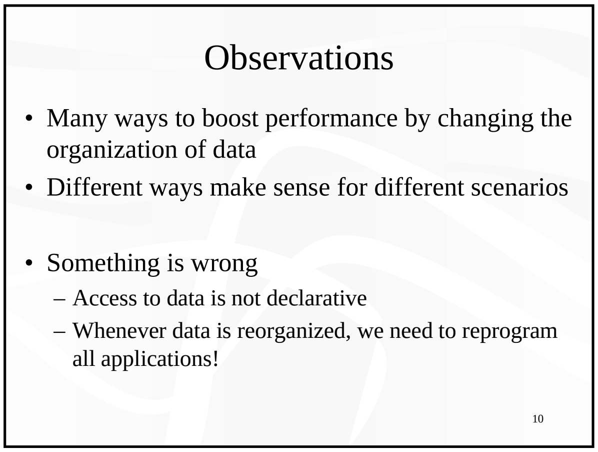


Performance problems

- Tens of thousands of accounts are not Homer’s

☝ Keep the accounts sorted by owner name
  - Change the script to do binary search
  - What happens when a new account is created?

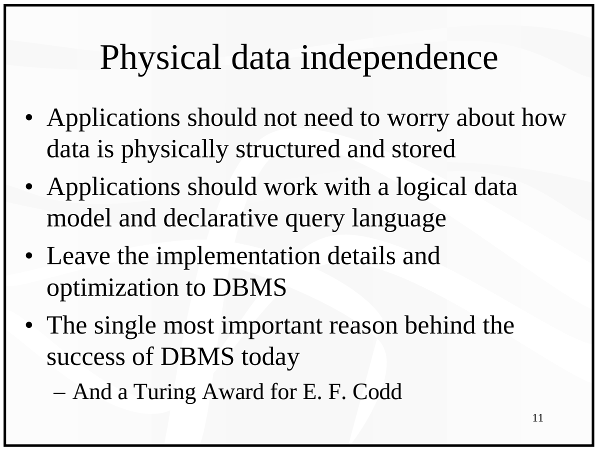
☝ Cluster accounts: Those owned by “A...” go into file A; those owned by “B...” go into file B; etc.
  - Change the script to decide which file to search
  - What happens when the query changes to: Which accounts have 0 balance?



Observations

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

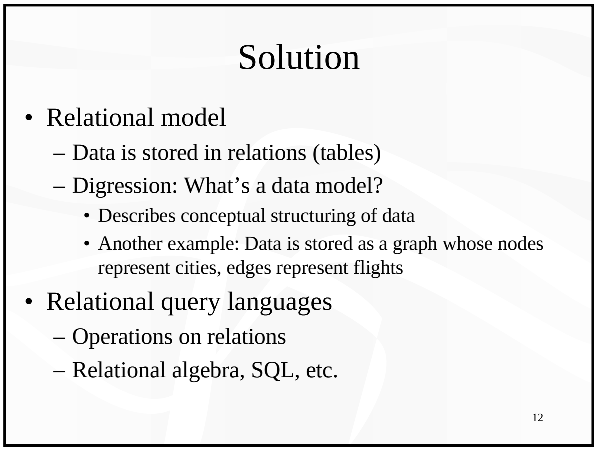
10



Physical data independence

- 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

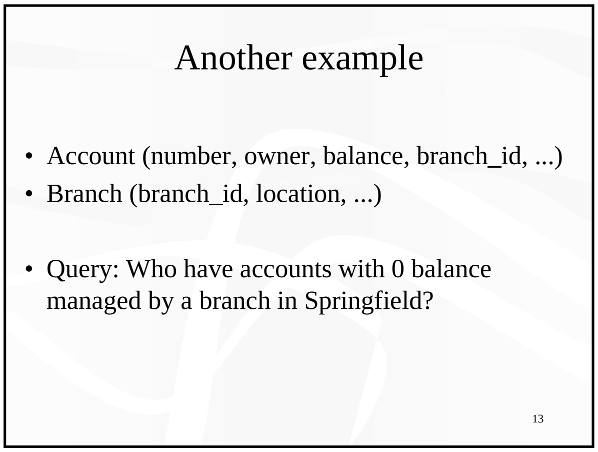
11



Solution

- Relational model
  - Data is stored in relations (tables)
  - Digression: What’s a data model?
    - Describes conceptual structuring of data
    - Another example: Data is stored as a graph whose nodes represent cities, edges represent flights
- Relational query languages
  - Operations on relations
  - Relational algebra, SQL, etc.

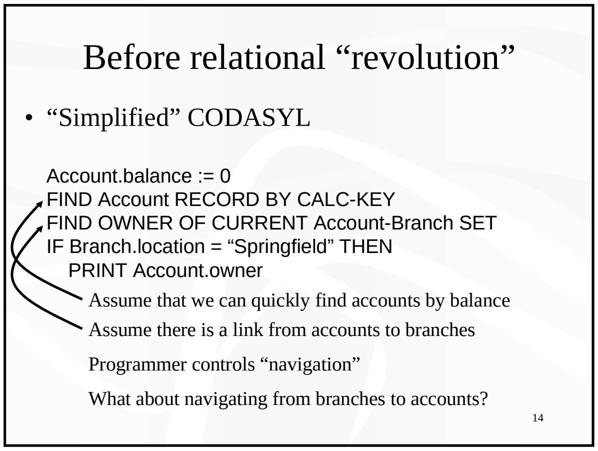
12



Another example

- 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”

- “Simplified” CODASYL

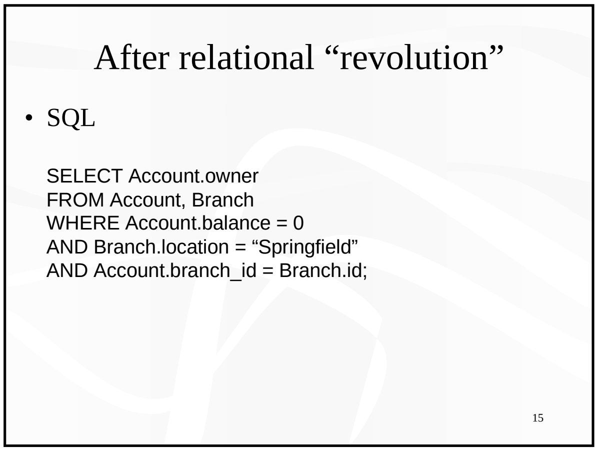
```
Account.balance := 0
FIND Account RECORD BY CALC-KEY
FIND OWNER OF CURRENT Account-Branch SET
IF Branch.location = “Springfield” THEN
   PRINT Account.owner
```

Assume that we can quickly find accounts by balance

Assume there is a link from accounts to branches

Programmer controls “navigation”

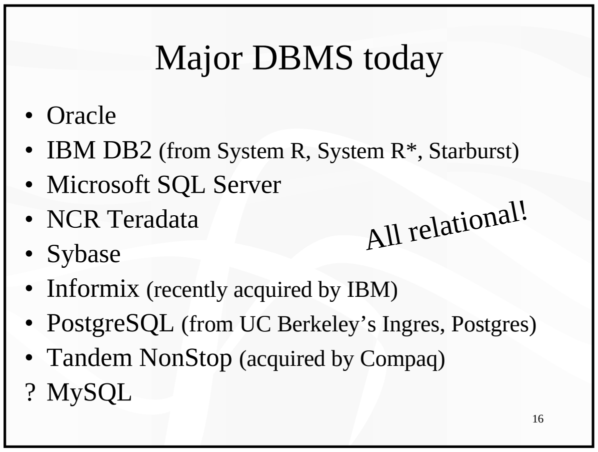
What about navigating from branches to accounts?



After relational “revolution”

- SQL

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



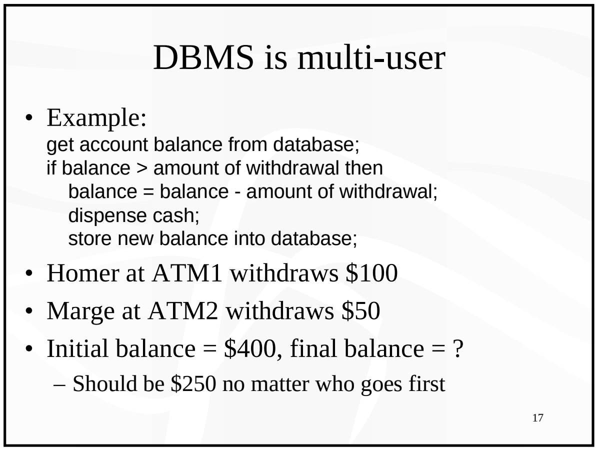
Major DBMS today

- Oracle
- IBM DB2 (from System R, System R*, Starburst)
- Microsoft SQL Server
- NCR Teradata
- Sybase
- Informix (recently acquired by IBM)
- PostgreSQL (from UC Berkeley’s Ingres, Postgres)
- Tandem NonStop (acquired by Compaq)

? MySQL

All relational!

16



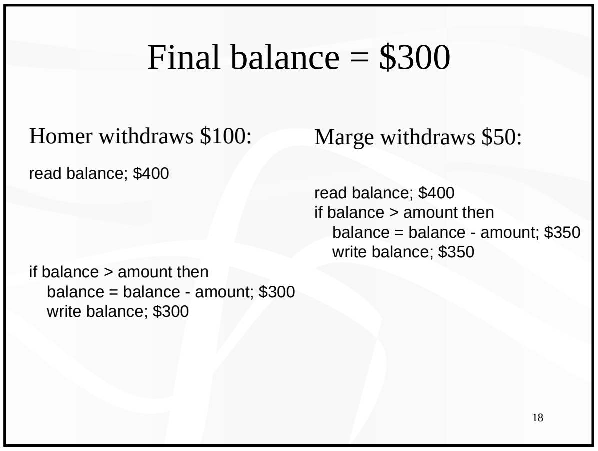
DBMS is multi-user

- 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

17

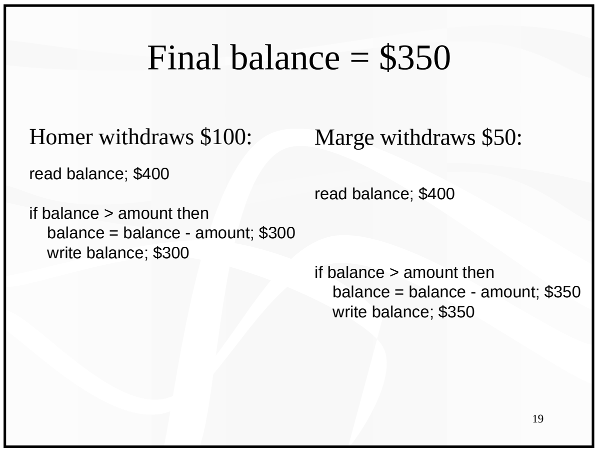


Final balance = $300

```
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
```

18



Final balance = $350

Homer withdraws $100:

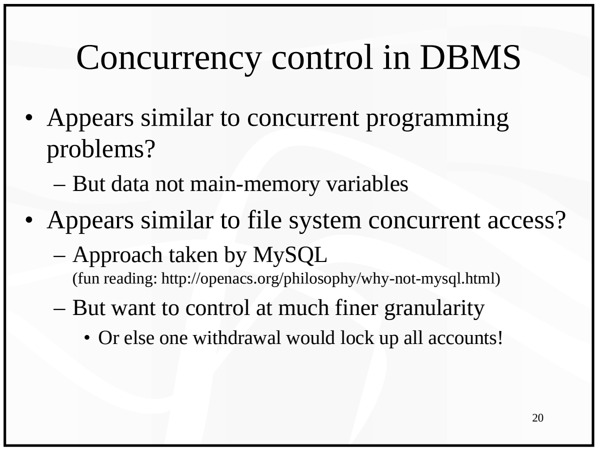
```
read balance; $400

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

Marge withdraws $50:

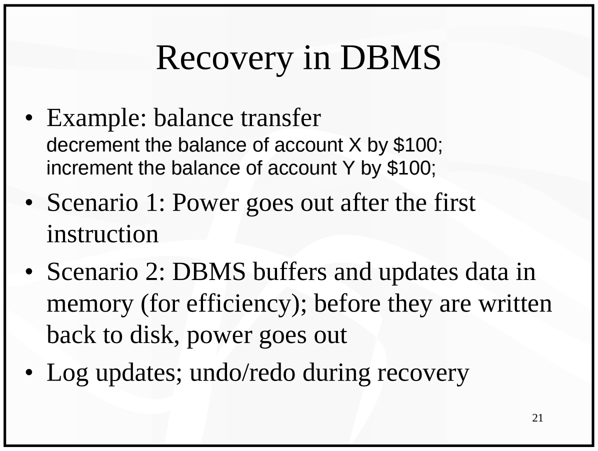
```
read balance; $400


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



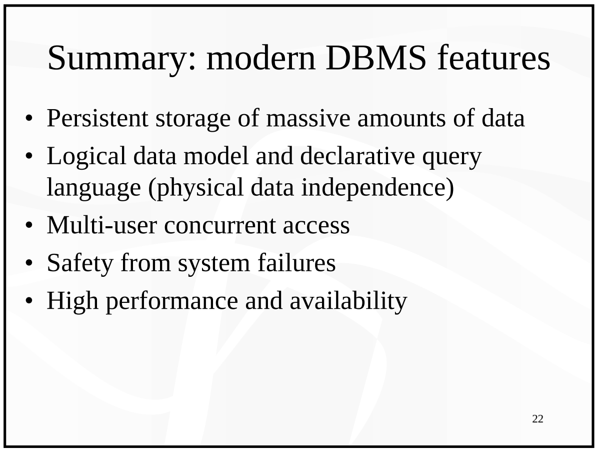
Concurrency control in DBMS

- 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!



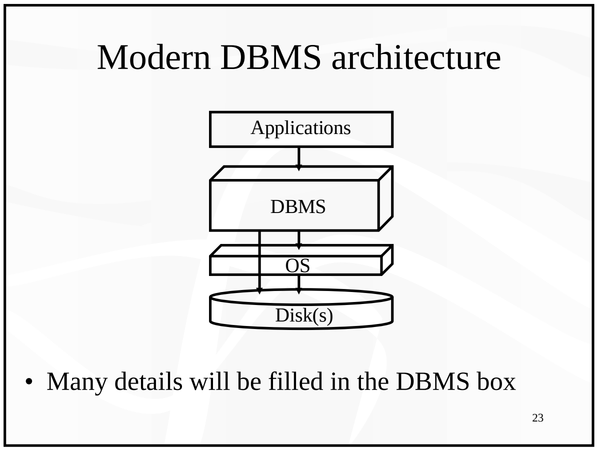
Recovery in DBMS

- 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



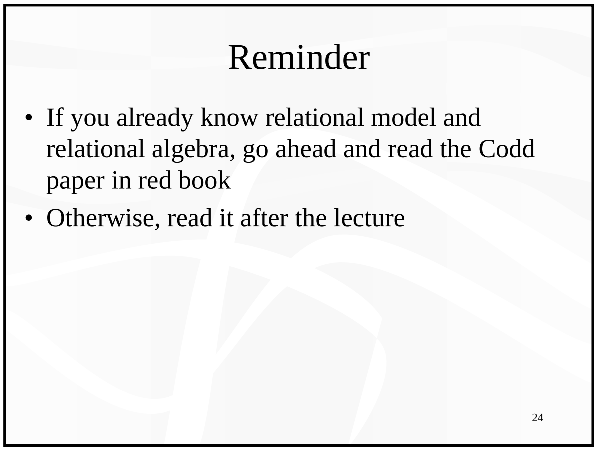
Summary: modern DBMS features

- Persistent storage of massive amounts of data
- Logical data model and declarative query language (physical data independence)
- Multi-user concurrent access
- Safety from system failures
- High performance and availability



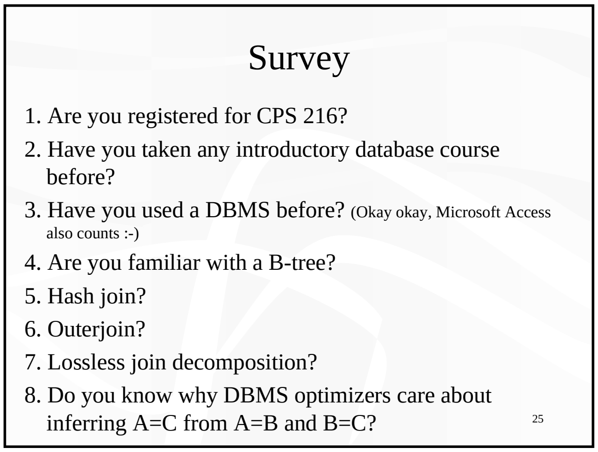
Modern DBMS architecture

- Many details will be filled in the DBMS box



Reminder

- If you already know relational model and relational algebra, go ahead and read the Codd paper in red book
- Otherwise, read it after the lecture



Survey

1. Are you registered for CPS 216?
2. Have you taken any introductory database course before?
3. Have you used a DBMS before? (Okay okay, Microsoft Access also counts :-)
4. Are you familiar with a B-tree?
5. Hash join?
6. Outerjoin?
7. Lossless join decomposition?
8. Do you know why DBMS optimizers care about inferring A=C from A=B and B=C?