Administrivia

- **Final Exam**
  - Monday, December 18 1230-330P
  - Bechtel Auditorium
  - Cumulative, stress end of semester
  - 2 cribsheets
- **Final Review Session**
  - Weekend before the exam?
  - Exact date TBA.
  - will blog the time and place

As you study...

- "Reading maketh a full man; conference a ready man; and writing an exact man."
  - Francis Bacon
- "If you want truly to understand something, try to change it."
  - Kurt Lewin
- "I hear and I forget. I see and I remember. I do and I understand."
  - Chinese Proverb.
- "Knowledge is a process of piling up facts; wisdom lies in their simplification."
  - Martin H. Fischer

Database Lessons to Live By

“If we do well here, we shall do well there:
I can tell you no more if I preach a whole year”
-- John Edwin (1749-1790)

Recall Lecture 1!!

- **Why Use a DBMS?**
  - Data independence and efficient access.
  - Reduced application development time.
  - Data integrity and security.
  - Uniform data administration.
  - Concurrent access, recovery from crashes.
- **Remind me again why we learned this stuff?**
  - Shift from computation to information
  - data sets get bigger and bigger
  - CS microcosm

Simplicity is Beautiful

- The relational model is simple
  - simple query language means simple implementation model
  - basically just indexes, join algorithms, sorting, grouping!
  - simple data model means easy schema evolution
  - simple data model provides clean analysis of schemas (FD’s & NF’s are essentially automatic)
  - Every other structured data model has proved to be a wash
  - What is the future of XML?
  - Why does the backend of web search look so much like a relational database?

Bulk Processing & I/O Go Together

- **Disks provide data a page at a time**
- **RDBMSs deal with data a set at a time**
  - sets usually bigger than a page
  - means I/O costs are usually justified.
  - much better than other techniques, which are "object-at-a-time"
- **Set-at-a-time allows for optimization**
  - can do bulk operations (e.g. sort or hash)
  - or can do things tuple-at-a-time (e.g. nested loops)
### Optimize the Memory Hierarchy
- **DBMS worries about Disk vs. RAM**
  - can spend a lot of CPU cycles thinking about how to best fetch off disk (e.g. query optimization, buffer replacement strategies)
  - I/O cost "hides" the think time
- **Similar hierarchies exist in other parts of a computer**
  - various caches on and off CPU chips
  - can play database-y games with these levels too, but there's less time to spare.

### Query Processing is Predictable
- **Queries take many predictable steps**
  - unlike typical OS workloads, which depend on what small task users decide to do next
- **DBMSs can use this knowledge to do MUCH better than the OS heuristics**
- **These lessons should be applied whenever you know your access patterns**
  - again, especially for bulk operations!
- **disclaimer**
  - I have done a lot of research based on the contrary
  - but remember: religion first

### Practical Algorithm Analysis
- **Because of the need for query cost estimation, database implementors understand the real costs of their main algorithms**
  - e.g. sorting is not $O(n \log n)$, it's linear
- **In many applications, the bottlenecks determine the cost model**
  - e.g. I/O is mostly what matters in DBs
  - this affects the practical analysis of the algorithm

### Indexing Is Simple, Powerful
- **Hash indexes easy and quick for equality**
  - worth reading about linear hashing in the text
- **Trees can be used for just about anything else!**
  - each tree level partitions the dataset
  - labels in the tree "direct query traffic" to the right data
  - "all" you need to think about in designing a tree is how to partition, and how to label!

### Not enough memory? Partition!
- **Traditional main-memory algorithms can be extended to disk-based algorithms**
  - partition input (runs for sorting, partitions for hash-table)
  - process partitions (sort runs, hash partitions)
  - merge partitions (merge runs, concatenate partitions)
- **Sorting & hashing very similar!**
  - have a look at the midterm solution

### Declarative languages are great!
- **Simple**: say what you want, not how to get it!
- **Should correctly convert to an imperative language**
  - Codd's Theorem says rel. calc. = rel. alg.
  - no such theorem for text search :-(
- **If you can convert in different ways, you get to optimize!**
  - hides complexity from user
  - accommodates changes in database without requiring applications to be recompiled.
- **Especially important when**
  - App Rate of Change << Physical Rate of Change
SQL: The good, the bad, the ugly

- SQL is very simple
  - SELECT..FROM..WHERE
- Well...SQL is kind of tricky
  - aggregation, GROUP BY, HAVING
- OK, OK. SQL is a big fat mess!
  - duplicates & NULLs
  - Subqueries
  - dups/NULLs/subqueries/aggregation together!
- Remember: SQL is not entirely declarative!!!
- But, it beats the heck out of writing (and maintaining!) C++ or Java programs for every query!

Query Operators & Optimization

- Query operators are actually all similar:
  - Sorting, Hashing, Iteration
- Query Optimization: 3-part harmony
  - define a plan space
  - estimate costs for plans
  - algorithm to search in the plan space for cheapest
- Research on each of the 3 pieces goes on independently! (Usually...)
- Nice clean model for attacking a hard problem

Database Design

- (And you thought SQL was confusing!)
- This is not simple stuff!!
  - requires a lot of thought, a lot of tools
  - there’s no cookbook to follow
  - decisions can make a huge difference down the road!
- The basic steps we studied (conceptual design, schema refinement, physical design) break up the problem somewhat, but also interact with each other
- Complexity here pays off in simplicity per record & per query
  - vs. files

CC & Recovery: House Specialties

- DBMSs are the last word on concurrency and reliability
  - transactions & 2-phase locking
  - write-ahead-logging
  - details are tricky, worked out over 20 years!
- Other folks have repeatedly dabbled in this, and usually don’t get it right!
  - be suspicious of new ideas for concurrency & fault tolerance
  - they often either don’t work, or provide weaker guarantees
    - sometimes without significant performance gains

Databases: The natural way to leverage parallelism & distribution

- The promise of CS research for the last 15 yrs:
  - There are millions of computers
  - They are spread all over the world
  - Harness them all: world’s best supercomputer!
- This was routinely disappointing
  - except for data-intensive applications (Dbs, Web)
- 2 reasons for success
  - data-intensive apps easy to parallelize & distribute
  - lots of people want to share data
  - fewer people want to share computation!
- The parallelism craze is BACK
  - Intel, AMD, etc need us to take advantage of parallelism
  - They have nothing else to do with all those transistors!
  - Incoming freshman will get this in 61A and through the curriculum

“More, more, I’m still not satisfied” — Tom Lehrer

- Grad classes @ berkeley
  - CS262A: a grad level intro to DBMS and OS research
  - CS286: grad DBMS seminar
  - read & discuss lots of OS & DBMS research papers
  - See evolution of different communities on similar issues
  - undertake a research project -- often big successes!
- CS194 "Hackatorium"
  - Agile software development lab
  - I.e. come hack with me
  - We will build a snazzy open source app on a "declarative networking engine", P2 (p2.cs.berkeley.edu)
But wait, there’s more!

- **Graduate study in databases**
  - Used to be rare (Berkeley + Wisconsin)
  - You are living in the golden age:
    - Berkeley (naturally!), Wisconsin, The Farm, MIT, Maryland, Brown, Cornell, CU, Duke, many others...
- **Tons of DB-related companies, almost EVERYONE hiring!**
  - Search companies
  - DB "elephants": IBM, Oracle, MS
  - Midstage DB startups: ANTs, Greenplum, Netezza...
  - Enterprise app firms: e.g., SAP, Siebel, Salesforce
  - DBA jobs
- **A note: ask for the job you want**
  - E.g. not just engineering -- sales, marketing, R&D, management, etc.

Parting Thoughts

- "Education is the ability to listen to almost anything without losing your temper or your self-confidence."
  - Robert Frost
- "It is a miracle that curiosity survives formal education."
  - Albert Einstein
- "The only thing one can do with good advice is to pass it on. It is never of any use to oneself."
  - Oscar Wilde