Database System Implementation Project

CS101 Section 3

Spring 2006-2007

Overview

- Opportunity to work on an interesting DB system implementation project
  - Doesn't have to be a relational database project
  - Can include object databases, XML databases, real-time databases, etc.
  - Should be focused on DB system implementation
- 9 unit course (1-8-0)
  - One hour of lecture/discussion each week
    - Some weeks will focus on project presentations
    - Lectures for other weeks, focusing on RDBMS impl.
  - Rest of time each week is focused on design and implementation of your project
    - Need a status update every week

Overview (2)

- General process:
  - Write a proposal and design document for project
  - Dive into research, then implementation and testing
  - At end of term, give a presentation/demo of your work
- Projects can be multi-person
  - Problem scope must be scaled appropriately to number of people
- Projects must have *measurable goals*
  - Tests that demonstrate correctness/functionality
  - Performance or load tests to demonstrate scaling
  - Try to include a simple demo of these tests in your final presentation

Project Schedule

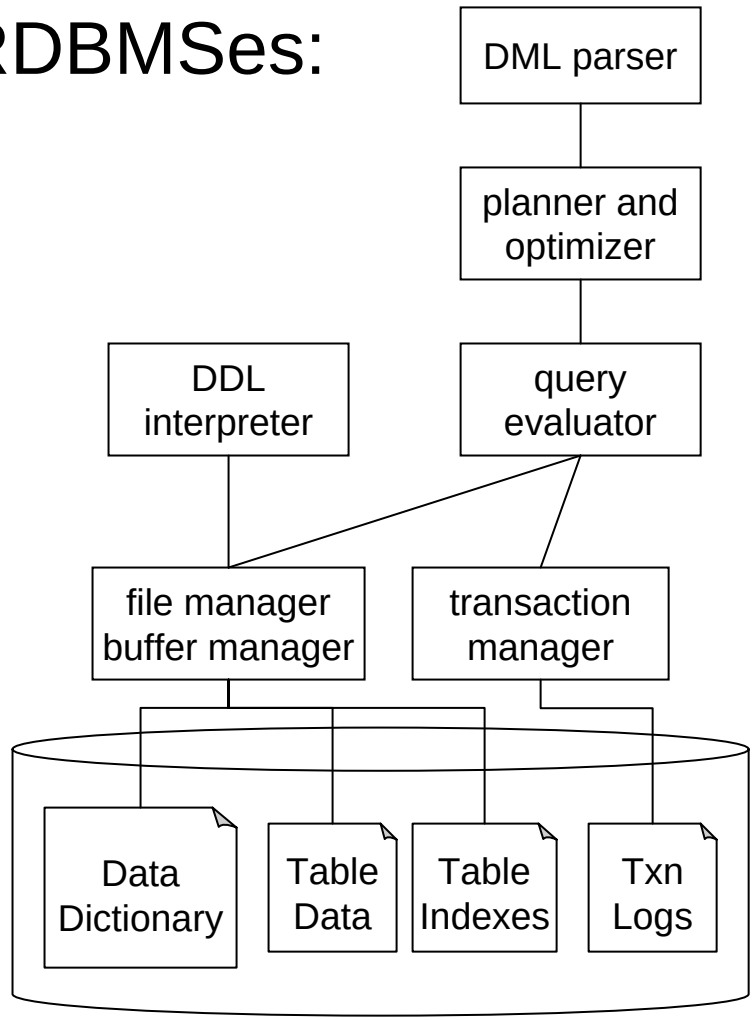
- Week 1: Write your project proposal
  - 1-2 pages describing what you want to do
  - Specify measurable goals!
- Weeks 2-3: Research, design document
  - Specify how you will implement your project
    - Language, platform, how you will demonstrate completion
  - Complete draft due at end of week 2
  - Revisions and schedule due at end of week 3
- Weeks 4-10: Implementation and testing
  - Each week: 5-minute status update in class
  - Week 5: brief presentation of project and status
  - Week 10: more in-depth presentation of results

Grading

- Course is on grades, but P/F is an option
- Final grade will be based on:
  - Design document
  - Mid-term presentation
  - Final presentation
  - Actual project code quality and functionality
- Success depends on you!
  - Lectures won’t necessarily focus on your project’s details
  - You will need to spend time researching your idea and designing your project
  - A good opportunity for you to practice these skills

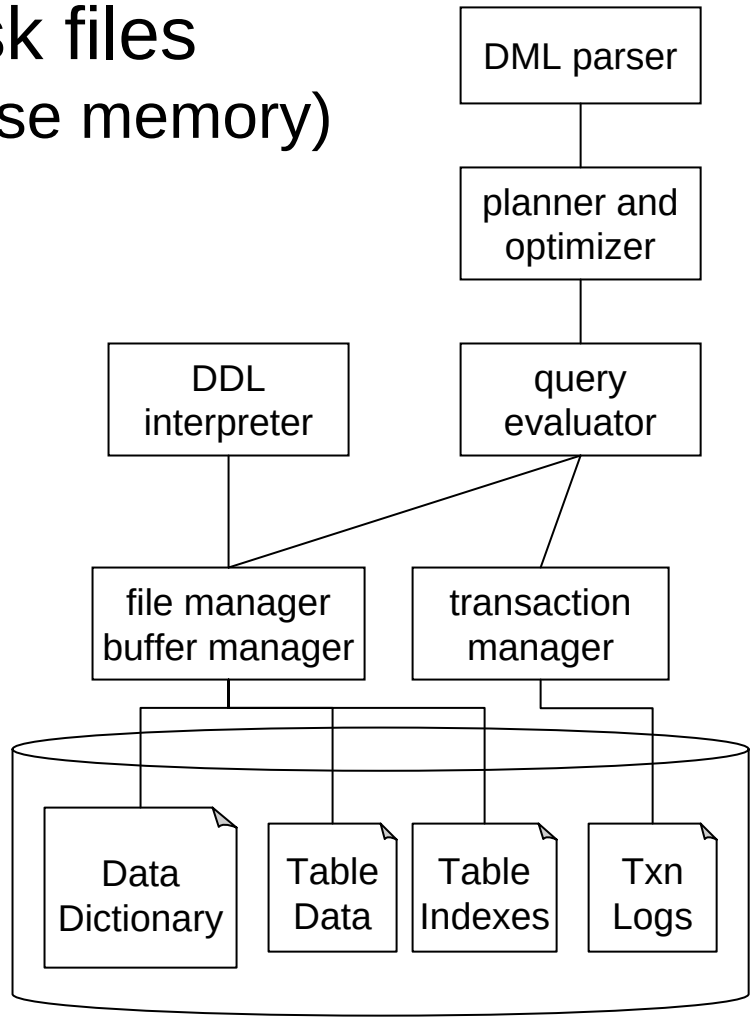
Relational Database Architecture

- A general architecture for RDBMSes:
  - Diagram is not complete
  - Other components as well
- Most DBs have separate paths for DDL, DML
  - DDL involves simple manipulation of table schemas, etc.
  - DML requires more complicated machinery



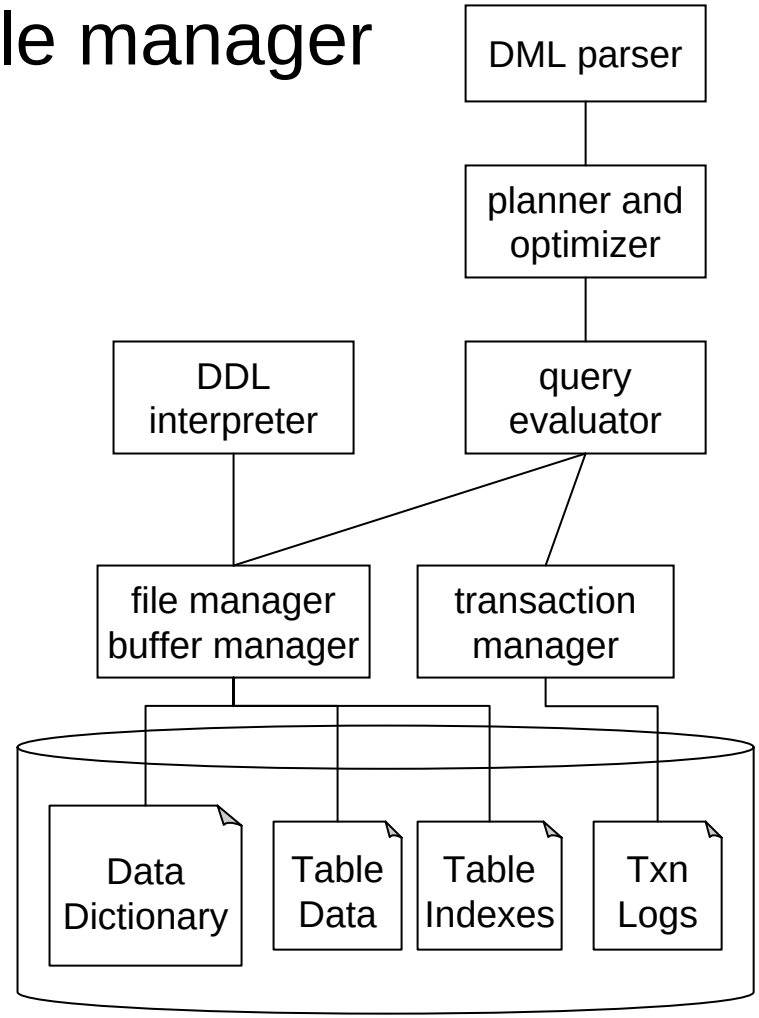
RDBMS Architecture (2)

- Data is usually stored in disk files
  - (Small DBMSes might only use memory)
  - File format is driven by specific purpose of file
  - Virtually all data files are read/written using pages
    - Page/block size usually between 4KB and 64KB
  - Data dictionary stored in same way as table data, to simplify management
  - Tuple abstractions, etc. are provided to query evaluator



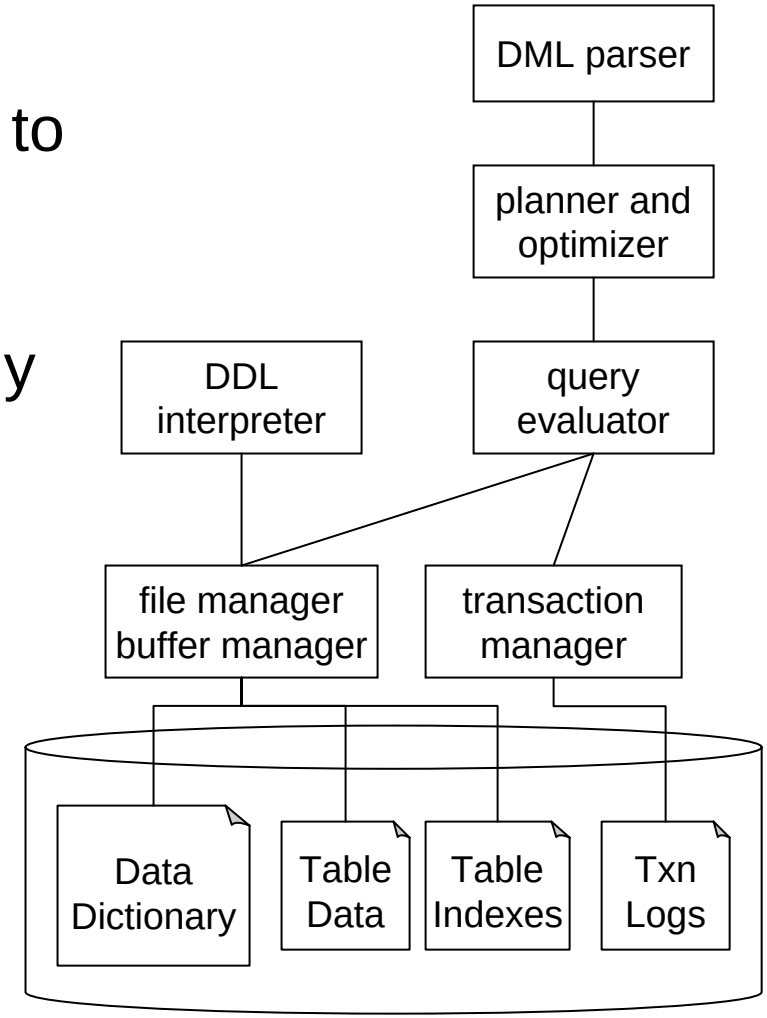
RDBMS Architecture (3)

- Data files manipulated by file manager
  - File access usually largest performance bottleneck
  - Buffer manager caches disk pages in memory to minimize file reads
  - Dramatically improves query performance
  - Also affects concurrency control and recovery!
  - File manager must provide a way to flush files to disk



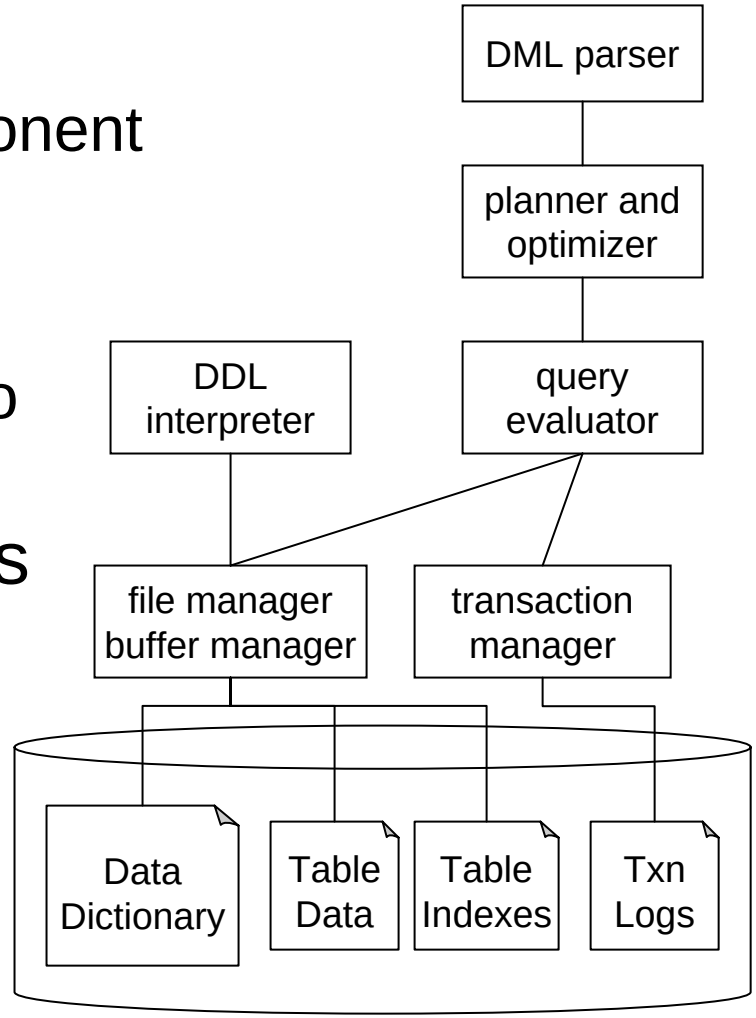
RDBMS Architecture (4)

- Query evaluation pipeline
  - Parsing SQL and converting to execution plan is simple
  - Planner/optimizer is critical!
  - Most SQL queries have many ways of being evaluated
    - Order of selects, projects, etc.
    - Join order, join strategies
    - Rewriting subqueries as joins plus grouping/aggregation
  - Must pick a good query plan, really quickly



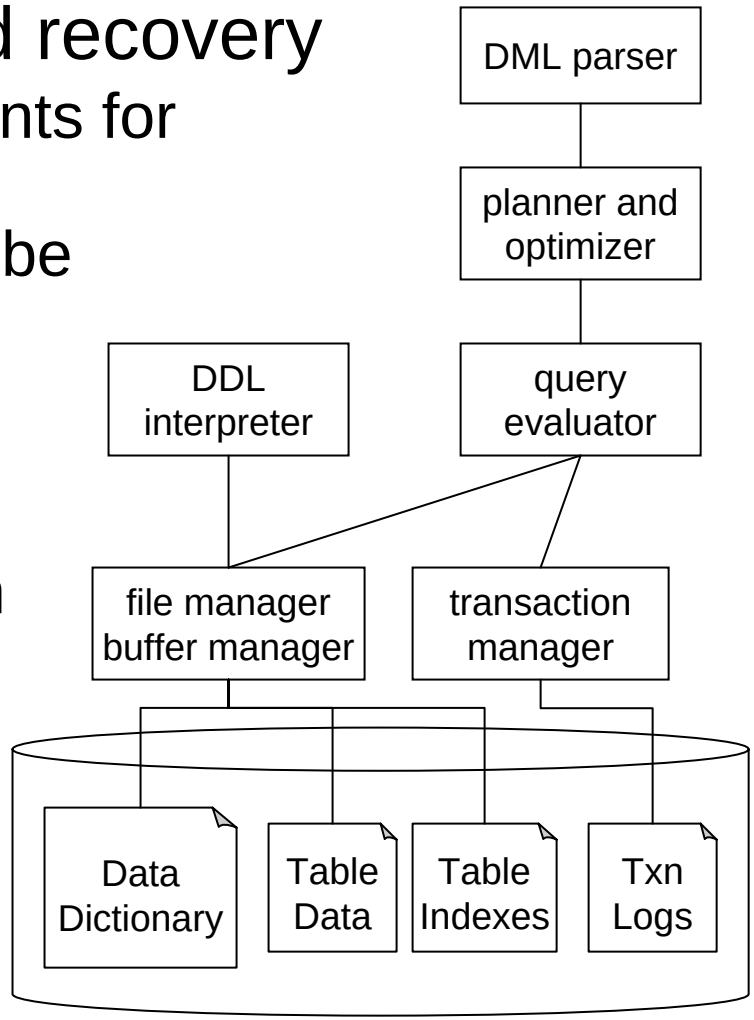
RDBMS Architecture (5)

- Query evaluation engine
  - Mostly straightforward component
- Complexities arise from:
  - Correlated subqueries
  - Derived relations that need to be *temporarily* materialized
- Handle updates and deletes using same pipeline
  - Recast updates, deletes as “select for update” and “select for deletion”



RDBMS Architecture (6)

- Transaction processing and recovery
  - DBMSes have *big* requirements for consistency and durability
  - In event of failures, DB *must* be restored to a consistent state
  - Transaction manager logs all [DML] operations
  - If a failure occurs, recovery manager can use transaction logs to restore a consistent state
  - Logging adds overhead, but this can be mitigated using checkpoints

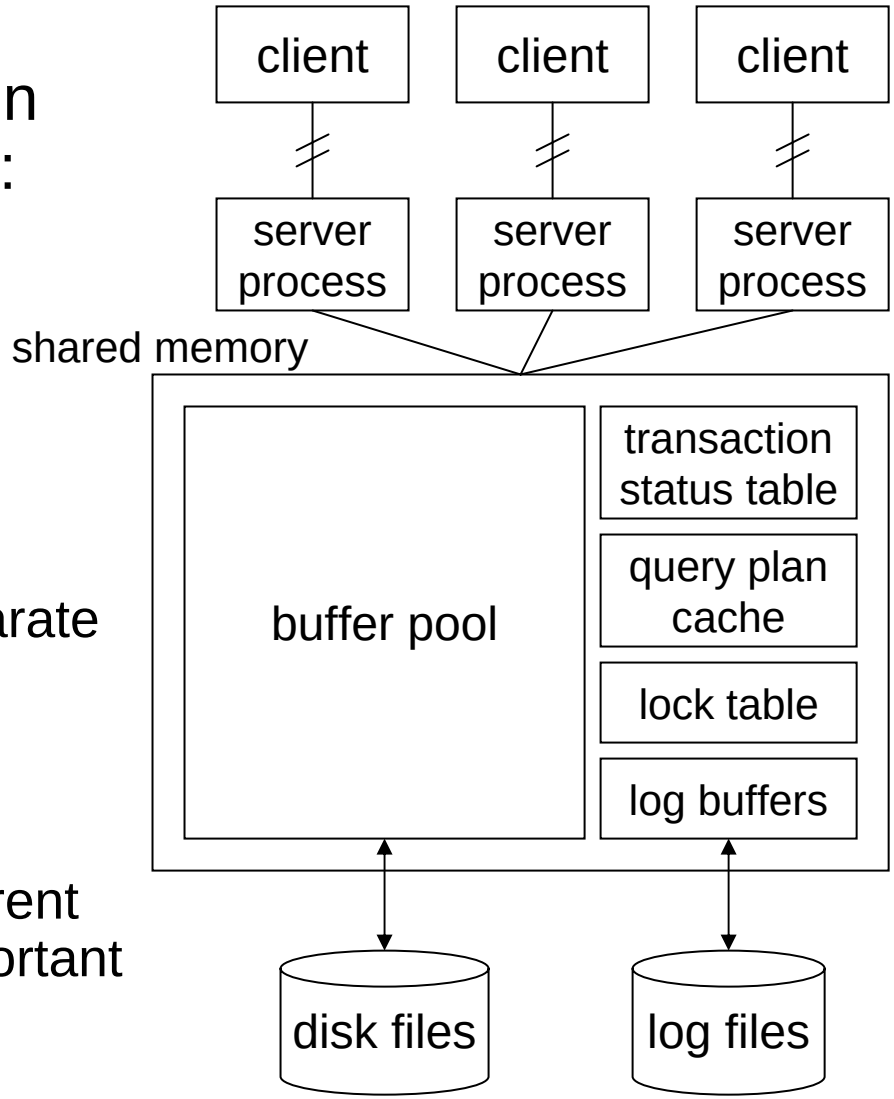


Concurrency Control

- Whether an RDBMS is single-user or multi-user has a large impact
- Single-user databases are much simpler
  - Only need to keep a single version of all data
  - No concurrency control, no lock management
  - Transaction processing is much simpler
- Multi-user databases require much more machinery
  - MVCC is most common storage technique
  - Concurrency control and lock management become critical
  - Transaction isolation must be managed carefully

Concurrency Control (2)

- Multi-user databases often follow a structure like this:
- All state must be shared across processes
- Server processes must coordinate reads/writes, transactions and locking
  - Sometimes done with separate processes, but not always
- Performance must be managed carefully
  - Resource usage in concurrent server processes also important



Example Project Ideas

- Write a file manager to store tuples on disk
  - Support variable-size tuples
  - Support single-version or MVCC records
  - Implement simple selects, inserts, updates, deletes against a single table
  - Write a simple buffer manager to cache page access
- Implement several kinds of index files
  - B-tree index, hash index
  - Provide APIs for common operations
    - e.g. start-scan, get-next, find, find-first, find-last, etc.
  - Handle index-update operations too!
    - add/delete index record, compact index, etc.

Example Project Ideas (2)

- Don’t need to limit yourself to only data files
  - Can write an in-memory database system that doesn’t require data files, storage formats, etc.
- Write a query executor for in-memory data
  - Parse simple SQL DML commands and generate execution plans to evaluate
  - Implement plan nodes for sequential scans, sorting, grouping/aggregation, joins
  - Use a heuristic-driven plan optimizer
  - Experiment with different strategies, measure performance

Example Project Ideas (3)

- Write a cost-based query planner/optimizer
  - Take simple, unoptimized execution plans as input
    - Will need an appropriate representation of query plan nodes
  - Output optimized execution plans, along with associated cost measure
  - Need to properly cost different plan nodes
    - Use (faked) table statistics to choose optimal plans
    - Take CPU, memory, disk requirements into account
    - Add support for distributed query planning; take network bandwidth into account
  - Integrate a mechanism for limiting search effort of optimizer

Example Project Ideas (4)

- Write a simple relational database with a front-end other than SQL
  - e.g. a Datalog-type language modeled after one of the relational calculi
  - Explore strategies for correct and efficient evaluation
- Could focus project on specific problem domain
  - Allowing easy statement of recursive or time-based queries
  - Easy statement of OLAP queries and computed results

Minibase

- Minibase is a simple RDBMS implementation
  - Specifically designed for educational use
    http://www.cs.wisc.edu/~dbbook/openAccess/Minibase/minibase.html
- Actually two RDBMS implementations
- C++ impl. provides many basic features
  - SQL parser, optimizer, buffer manager
  - Heap files for tuple storage
  - B+ tree index implementation
- Java impl. provides only lower-level features
  - No SQL parsing or planning/optimization provided

Minibase (2)

- Could build a project on top of Minibase
  - Implement your own version of one of its components
    - Implement a buffer manager
    - Implement several join algorithms that actually work with disk files
  - Add some new features
    - Build a SQL front-end for Java Minibase impl.
    - Implement transaction support using a write-ahead log and checkpoints

Other Project Ideas

- Could also take an existing database system and enhance it in some way
  - Actually modify the internals of the DBMS
  - Build an external component that integrates with the DBMS to extend its functionality
- Look at open-source database impls
  - Apache Derby (formerly IBM Cloudscape)
  - PostgreSQL, MySQL
  - HSQLDB
  - …

Next Steps

- Monday, April 2:
  - Let me know if you are taking the class
  - If so, briefly describe your project ideas
- Wednesday, April 4:
  - 1st draft of project proposal/design doc is due
- Should include:
  - Clear statement of project focus
  - General implementation details, e.g. language, platform, file-based vs. in-memory, other details
  - Measurable goals you intend to achieve
  - Include references to papers, books, websites you will use to guide you