Announcements (April 23)
- Homework #4 will be graded by Saturday
  - Sample solution available today
- Verify the accuracy of your scores in Blackboard and let me know of any problem before the final
  - Homework assignments, midterm, reviews, presentation

Announcements (cont’d)
- Final exam next Monday (April 26)
  - 2-5pm, in this room (D243 LSRC)
  - Comprehensive (everything up to today’s lecture, with emphasis on the second half of the course, and materials exercised in homework assignments)
  - Open book, open notes; no time pressure
  - Sample final and solution available today (note the difference materials covered in last year’s CPS216)
- Project demos Tues./Wed. after the final
  - Email confirmation of schedule will be sent later today
  - Remember that report is due before the demo

Pre-midterm: basics
- Relational model/algebra → physical data independence
  - Really made query optimization flourish
- SQL: NULL and three-value logic, bag versus set semantics, subqueries, grouping and aggregation → nifty features, mess for optimizers
  - Recall query rewrite tricks for preserving duplicate, avoiding the count bug, and magic decorrelation
  - Use query rewrite to get back to the simplicity of relational algebra

Pre-midterm: basics (cont’d)
- More SQL
  - Views → logical data independence
    - Materialized views → reintroduce redundancy to improve performance
  - Constraints → the more you know the better you can do
    - Did not cover semantic query optimization
  - Triggers (ECA) → “active” data
    - Did not cover scalable trigger processing (related to multi-query processing for continuous queries)

Pre-midterm: physical data organization
- Storage hierarchy (DC vs. Pluto)
  - Count I/O’s
  - Get as much useful info as possible with each long trip
  - Do other things while waiting
- Disk performance → sequential beats random
- Data layout
  - Record layout (handling variable-length fields, NULL’s)
  - Block layout (NSM, DSM, PAX)
    - Inter-/intra-record locality
### Pre-midterm: physical data organization (cont'd)

- **Access paths**
  - Primary versus secondary indexes
  - Tree-based indexes: ISAM, B⁺, B, R, R*, R⁺, GiST
  - Hash-based indexes: extensible, linear
  - Text indexes: inverted lists, signature files (and bit-sliced ones), suffix array, trie, suffix tree, Patricia trie, Pat tree
  - Variant indexes: value-list(bitmap), projection, bit-sliced indexes, join indexes
  - Reintroduce redundancy to improve performance
  - Fundamental trade-off: query versus update cost

### Pre-midterm: query processing

- **Scan-based algorithms**
- **Sort- and hash-based algorithms (and their duality)**
- **Index-based algorithms**
- **Pipelined execution with iterators**
  - Blocking and non-blocking operators
- **Buffer management**
  - Per-query, per-table policy is ideal
  - The more you know the better you can do

### Review: XML basics

- **Data model: well-formed vs. valid (DTD ≈ schema)**
- **Query languages**
  - XPath: (branching) path expressions (with conditions)
  - XQuery: FLWR, subqueries in return (restructuring), quantified expressions, aggregation, sorting
  - XSLT: structural recursion with templates
- **Programming: SAX (one pass) vs. DOM (in memory)**

### Review: representing XML

- Flat files and CLOB do not really exploit the structure of XML
- **Schema-oblivious approaches**
  - Node/edge representation
  - Interval-based representation (left, right, level)
  - Path-based representation (labeled path, Dewey order)
  - Sequence-based representation (ViST)
- **Schema-aware approach**
  - Inlining choice for *, +, and shared elements in DTD
  - Less flexible and harder to reformulate queries, but queries are more efficient → the more you know the better you can do

### Review: processing XML

- **Finite state machines (Niagra, YFilter)**
- **Node/edge representation**
  - Naturally leads to navigational processing
  - Path expression steps → equality joins
    - Top-down, bottom-up, hybrid, … correspond to different join orders
- **Interval-based representation**
  - Naturally leads to structural join processing
  - Path expression steps → containment joins (great for asc/desc)
    - Join ordering? Less of an issue because it can be processed as a multi-way join on the same attribute
  - Stacks are your best friend; remember XML intervals don’t overlap
  - A mixed-mode approach may be best
  - Everything comes down to joins!

### Review: indexing XML

- **Basic indexes**: inverted lists for tag names, value indexes, back pointers to parents, etc.
- **Index for interval-based representation**
  - Example: XR-tree (B⁺-tree augmented with stab lists at internal nodes) for finding ancestors
- **Index for path-based representation**
  - Example: IndexFabric (based on Patricia trie)
- **Index for sequence-based representation**
  - Example: ViST
  - Path expression → (non-contiguous) subsequence matching
  - Use a trie to score sequences, encoded using intervals to support skipping
- **Structural summary indexes for graphs**
  - Examples: DataGuide (DFA) and 1-index (NFA)
  - Still plenty of room for improvement
Review: query optimization or "goodification"?

- **Heuristics**: push selections down; smaller joins first
  - Reduce the size of intermediate results
- **Cost-based**
  - Query rewrite
    - Apply relational algebra equivalences to SPJ blocks
    - Merge blocks to get a bigger search space
  - Cost estimation
    - boils down to estimating the size of intermediate results
    - Use statistics (e.g., histograms) → fundamental trade-off: cost versus accuracy
  - Search
    - Dynamic programming (+ interesting orders), randomized search, genetic programming, etc.