Midterm Review

CPS 216

Advanced Database Systems

Announcements (February 26)

- Homework #2 due today
  - Sample solution will be available by Monday
- No reading assignment for the coming week
- Midterm exam next Thursday in class
  - Everything before XML
  - Open book, open notes
- Project milestone 1 due next Friday
  - See project description for what and how to submit

Review: basics

- Relational model/algebra → physical data independence
- Design theory (FD's, BCNF) → help eliminate redundancy
- SQL
  - `NULL` and three-value logic → nifty feature, big mess
  - Bag versus set semantics
  - Subqueries, grouping and aggregation → which features add more expressiveness?
  - Views → logical data independence
    - Materialized views → reintroduce redundancy to improve performance
  - Constraints → the more you know the better you can do
- Covered in recitations (will not be in the exam):
  - Triggers (ECA) → "active" data
  - Transactions and isolation levels

Review: physical data organization

- Storage hierarchy (DC vs. Pluto)
  - → Count I/O's
  - → Get as much useful info as possible with each long trip
- Disk performance → sequential beats random
- Data layout
  - Record layout (handling variable-length fields, `NULL`'s)
  - Block layout (NSM, DSM, PAX)
    - → Inter-/intra-record locality

Review: 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

Review: 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