Announcements

• Webquiz 7 due date extended to tomorrow

• Extra office hours tonight 6pm to 8pm on 4th floor CSE
Final Exam

- Thursday, Dec 13  2:30 - 4:20
- This room
- Closed books, no phones, no computers
- Allowed 2 pages of notes (both sides of each)
- Primary focus on the second half
Course Topics

1. Relational Data
2. Database Design & Implementation
3. Concurrency and Transactions
4. Semi-structured Data
5. Parallel Big Data Systems
When to Use a Database

• Of all the tools we know, how do we pick?
• Depends on use case and role
• Easier to implement in database first than reverse engineer
Role: Scientist

• Data type: single csv file
  – Probably don’t need database unless you love SQL

• Data type: many small csv files
  – Good time to use SQLite, easier to add data to one place

• Data type: multiple table catalog
  – Consider more powerful database like PostgreSQL or SQL Server
Role: Industry Data Analyst

• Very small company: see scientist case
  – Will have to think about database design

• Large company: definitely need database
  – System: whatever they tell you to use
    • Probably the parallel database system de jour (SQL or NoSQL)
  – Often very concerned with speed
    • Can take hours to run a query
Role: Industry Application Developer

- Database design
  - Single node or parallel database?
  - SQL or NoSQL/MapReduce?
  - Indexes

- Making app work with database
  - Transactions
  - Query execution
Relational Data
1a. Relational Data Model

• Tables with **schemas**
  – First normal form: no nested tables
  – types for attributes
  – primary and foreign keys
  – other constraints

• **bag semantics**
1b. Relational Queries

- Relational query = expressible in standard RA
- Simple SELECT-FROM-WHERE with no joins or aggregates is:
  - Expressible with nested-loop semantics
  - Always monotone
- Datalog adds recursion
SQL

• CREATE TABLE …
  – PRIMARY KEY, FOREIGN KEY
  – Constraints UNIQUE, NOT NULL
• CREATE [CLUSTERED] INDEX … ON ...
• INSERT INTO …
• UPDATE … SET … WHERE ...
• DELETE FROM … WHERE …
SQL (cont.)

• SELECT …
  – JOINs: inner vs outer, natural
  – GROUP BY, sum, count, avg, etc.
  – ORDER BY

• BEGIN TRANSACTION

• COMMIT / ROLLBACK
Database
Design & Implementation
2a. DB Design Process

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details
Physical Schema
2a. DB Design Process

• E/R Diagrams
  – Entity sets, relations, & subclasses
  – Map each to relations
    • multiple ways to do this (many-many, one-many)
  – Design principles:
    • model accurately
    • neither too few nor too many entities
2a. DB Design Process

• Constraints
  – key, referential & other constraints

• Normalization
  – Eliminates anomalies
    • redundancy, update, and deletion anomalies
  – Occur from “bad” functional dependencies (FDs that aren’t superkeys)
  – Apply BCNF decomposition to remove them
3-Tiered Architecture

Connection (e.g., JDBC)

HTTP/SSL
2a. Storage & Indexing

• B+ tree & hash indexes
  – B+ tree index is sorted, best for range queries

• clustered vs unclustered
  – clustered always speeds up query
    but only one index per table can be clustered
  – unclustered only speed up selections if <1% tuples match
Query Evaluation Steps

1. **Parse & Check Query**
   - SQL query
   - Check syntax, access control, table names, etc.

2. **Decide how best to answer query: query optimization**
   - Logical plans, Physical plans

3. **Query Execution**

4. **Return Results**

CSE 414 - Autumn 2018
3b. Query Optimization

• main cost is disk access
• many logical plans, many physical plans
  – logical plans are RA expressions with desired result
  – physical plans include e.g. choice of join algorithm
    • E.g. block nested loop join and index nest look join
• cost of many operations depends on selectivity
• optimization problem is hard
• realistic goal is to avoid really bad plans
Concurrency and Transactions
4c. Transactions

• goal to allow many clients to run simultaneously
  – OLTP workload: lots of clients with small read/writes
• need to provide ACID properties
  – Atomic
  – Consistent
  – Isolated
  – Durable
4c. Transactions II

- isolation achieved through serializable schedules
  - serializable means same behavior as a serial schedule
  - conflict serializable means non-conflicting read/writes can be swapped to make schedule serial
    - stronger than (so implies) serializable

- locks ensure conflict serializability if 2PL used
4c. Transactions III

- 2PL: all locks proceed unlocks
- Strict 2PL: must do all unlocks at commit/rollback time
  - needed for isolation if txns can roll back
- need more to prevent phantom rows
  - phantom is a new row that shows up in a table
  - predicate locks are one solution (but expensive)

- default isolation level is usually not serializable
  - faster perf but harder to write app (i.e., bugs likely)
Semistructured Data
4a. Semistructured Data Model

- tree structured data: JSON, XML, etc.
- data is self-describing
  - so schema is not necessary
- easy to map relation to JSON but not opposite
Systems for Big Data
5a. NoSQL Systems

- goal to support heavy OLTP workloads
- provides simplified data model
  - key-value pairs, documents, or extensible records
- limited support for transactions
  - usually pair/document/record level
  - (some support for record groups… all on one node)
5b. Parallel Processing Systems

- for OLAP workloads (big reads, no txns)
- Goal is linear speed up or scale up
- MapReduce
  - programming model is one-to-many map function, shuffle sort (grouping), one-to-many reduce function
  - no built-in RA operators
  - stores intermediate data on disk
  - deals with stragglers by running backup map tasks
5c. Parallel Relational Databases

- Partition data across nodes (hash, range, etc.)
- Query evaluation
  - only one new element: reshuffle
    - move tuples to nodes based on values in certain columns
    - basically same as shuffle sort of MapReduce
    - use to implement all extended RA operations
  - new problem: skewed data
Things NOT on the final exam

- Datalog
- SQL++ queries (but there will be NoSQL conceptual questions)
- Join algorithms that were not in the slides
  - (no sort-merge joins but yes nested loop join)
- The difference between B+ tree vs. Hash indexes, can always assume B+ tree
- Isolation levels
- Writing Java code