**CS 133: Databases**

Spring 2017
Lec 26 – 4/27
Data Analytics & Course Overview
Prof. Beth Trushkowsky

**Goals for Today**

- Discuss decision support analytics queries and systems
- Brief overview of course
- Course evaluations

**Data Analytics and Decision Support**

- **Idea**: current and historical data to identify useful patterns and support business strategies
- **Complex, interactive, exploratory analysis of data**
  - Large datasets
  - Data integrated from across all parts of an enterprise
  - Data is fairly static

- **OLAP**: on-line analytical processing
  - In contrast to **OLTP** (on-line transactional processing)

**OLAP vs. OLTP**

- **OLTP**
  - Update-heavy
  - Short, simple transactions
  - Goal: transaction throughput

- **OLAP**
  - Mostly reads
  - Longer, complex queries for analysis and decision-making
  - Goal: fast queries
Data Integration

- Data may reside in many distributed, heterogeneous OLTP sources
  - Sales, inventory, customer, ...
  - NC branch, NY branch, CA branch, ...

- Need to support OLAP over integrated view of the data

- Possible approaches to integration
  - **Eager**: integrate in advance and store the integrated data in a data warehouse
  - **Lazy**: integrate on demand; process queries over distributed sources—the approach of mediated or federated systems

---

Example: Car Sales Schema

```
Cars(serialNo, make, model, color)
Dealers(name, city, state, phone)
Date(date, day, week, quarter, month, year)
Sales(serialNo, date, dealer, price)
```

---

Star Schema in Relational OLAP (ROLAP) System

- Fact table **BCNF**; dimension tables possibly **denormalized**
  - Dimension tables are small; updates/inserts/deletes are rare.... anomalies less important than performance

- Star Schema

---

A Multidimensional View

- Example car sales schema:

```
Cars(serialNo, model, color)
Dealers(name, city, state, phone)
Date(date, day, week, month, year)
Sales(serialNo, date, dealer, price)
```

---
**Dicing the Cube**

- Can think of partitioning the raw data cube along each dimension at some level of granularity

- A choice of partition for each dimension “dices” the cube

**Example: Data Analysis**

- Suppose *Mazda3* model is not selling as well as anticipated

- **Query:** which colors not doing well?

  ```sql
  SELECT color, SUM(price)
  FROM Sales NATURAL JOIN Cars
  WHERE model = "Mazda3"
  GROUP BY color;
  ```

**Slicing the Cube**

- Idea: want info about a fixed *slice* of the data

- In general, in SQL:
  - **Dice:** GROUP BY
  - **Slice:** WHERE

**Exercise 2 (a-c)**

(a)  
SELECT color, month, SUM(price)  
FROM Sales, Cars, Days  
WHERE Sales.serialNo = Cars.serialNo  
AND Sales.date=Days.date  
AND model = "Mazda3"  
GROUP BY color, month;

(b)  
SELECT dealer_name, month, SUM(price)  
FROM Sales, Cars, Days  
WHERE Sales.serialNo = Cars.serialNo  
AND Sales.date=Days.date  
AND model = "Mazda3"  
AND color = "red"  
GROUP BY month, dealer_name;

(c)  
SELECT dealer_name, year, SUM(price)  
FROM Sales, Cars, Days  
WHERE Sales.serialNo = Cars.serialNo  
AND Sales.date=Days.date  
AND model = "Mazda3"  
AND color = "red"  
AND (year = 2016 OR year = 2017)  
GROUP BY year, dealer_name;
OLAP Queries

• A common operation is to aggregate a measure over one or more dimensions.

• **Roll-up**: Aggregating at coarser granularity, e.g., higher level in dimension hierarchy.

• **Drill-down**: The inverse of roll-up

---

Analyzing Big Data: Current Trends

• **Motivation**
  – Expensive ROLAP and MOLAP systems not for everyone
  – Desire to analyze semi-structured or unstructured data

• **Big Data** rampant!
  – E.g., data sets generated by some of the applications backed by NoSQL systems
  – Sensor data, tweets, etc.

• **Trend**: many people using **MapReduce/Hadoop** for Big Data Analysis
  – Scalability and commodity hardware

---

MapReduce Programming Model

• **Data type**: key-value *records*

• **Map function**:
  \[(K_{in}, V_{in}) \rightarrow \text{list}(K_{intermediate}, V_{intermediate})\]

• **Reduce function**:
  \[(K_{intermediate}, \text{list}(V_{intermediate})) \rightarrow \text{list}(K_{out}, V_{out})\]

---

Example: Word Count

```python
def mapper(line):
    foreach word in line.split():
        output(word, 1)
def reducer(key, values):
    output(key, sum(values))
```

---

Specialized MOLAP and ROLAP systems may store pre-aggregated data (materialized views)

Open-source version of Google’s MapReduce

Thanks Mike Franklin & Matei Zaharia

Input could just be a unit, like line of text input
Word Count Execution

Input
the quick brown fox
the fox ate the mouse
how now brown cow

Map
the, 1
brown, 1
fox, 1

Map
the, 1
fox, 1
the, 1

Map
how, 1
now, 1
brown, 1

Map
ate, 1
cow, 1
mouse, 1

Reduce
brown, 2
fox, 2
how, 1
now, 1
the, 3

Reduce
fox, 2
how, 1
now, 1
the, 3

Reduce
ate, 1
cow, 1
mouse, 1
quick, 1

Output
brown, 2
fox, 2
how, 1
now, 1
the, 3

Pokémon or Big Data?
https://pixelastic.github.io/pokemonorbigdata/

Hadoop is Big Data!
Hadoop is a distributed system for counting words.

Why Use a DBMS?

• Data independence and efficient access
  – Declarative language: say what you want, not how to get it!
  – Hides complexity from user
  – Accommodates changes in database without requiring applications to be recompiled

• Data integrity and security

• Concurrent access, recovery from system crashes.

Final Exam: Logistics

• Take-home exam
  – Available to you Friday April 28th, 4pm
    (printed, Olin hallway cubbies)
  – Due to my office (Olin 1267) at or before:
    Seniors May 4th at 2pm
    Non-seniors May 1-0th at 5pm

• Two 8.5x11, double-sided note sheets
  – You can use your note sheet from midterm as one of the two
  – No other resources

• 3-hour timed exam
Simplified RDBMS Architecture

Optimize the Memory Hierarchy

• DBMS worries about Disk vs. Memory
  – Can spend a lot of CPU cycles thinking about how to best fetch data from disk
    • e.g., query optimization, buffer replacement strategies
  – I/O cost “hides” the CPU think time

Practical Algorithm Analysis

• Due to need for query cost estimation, DBMS developers understand the real costs of their main algorithms
  – e.g., external sorting

• In many applications, the bottlenecks determine the cost model
  – E.g., I/O is mostly what matters in DBs
  – Affects the practical analysis of the algorithm

• Distributed setting forces us to rethink some of these costs!

Design and Tradeoffs

• ACID transactions!
  – Concurrency and reliability
  – Two-Phase Locking and Isolation tradeoffs

• Database design (modeling real-life application)
  – Tools like BCNF normalization to help avoid anomalies
  – Heuristics and tradeoffs
Course Objectives

• Provide a solid background in database management system design principles

• Promote understanding of these principles through hands-on exercises implementing the internals of a relational database management system

• Further develop students' ability to reason about algorithm and software design, optimization, and tradeoffs generally applicable in computer science

Possible Topics on Final

• Cumulative-ish
  – Topics we covered earlier still relevant (e.g., hash & tree indexes, estimating cost in I/Os)
  – Won’t focus on nitty gritty from before midterm (e.g., linear vs extendible hashing)

• Query Optimization
• Transactions and ACID
• Database design
• ORDBMS, Distributed DBMS and NoSQL, OLAP (high-level)

• General themes
  – Reasoning about cost and tradeoffs
  – Consistency and correctness with concurrent access and failures

Query Optimization

• Query
  ⇒ relational algebra tree
  ⇒ logical plan
  ⇒ physical plan

• Unit of optimization: query block

• Logical plan
  – Relational algebra equivalences
  – Outer vs. inner relation in joins
  – Query plan tree shape: bushy, linear, deep

Query Optimization

• Choosing physical plan
  – Enumerate plan space
    • Join permutations and orders
    • System R choices
  – Estimate cost of plan
  – Picking cheapest
    • Dynamic programming algorithm (idea)
    • Interesting orders

• Cost estimation
  – Operator algorithm cost
    • Estimating cost of different join algorithms
  – Operator result size estimation
    • Selectivity/Reduction Factor, statistics, histograms
    • Using indexes
ACID Transactions

- Transactions, how to achieve ACID
- Isolation (I)
  - Schedules: serializable, conflict-serializable, etc.
  - Anomalies from interleaved actions, conflicting actions
  - Locking, lock granularity and compatibility, deadlock detection and prevention
  - 2PL vs Strict 2PL, cascading aborts
  - Optimistic concurrency control, backwards validation algorithm
- Recovery (A and D)
  - Steal vs. force and implications on UNDO/REDO
  - Write-Ahead-Logging
  - ARIES recovery algorithm

Database design

- E/R modeling (general idea)
  - Entities, relationships, weak entities
  - Capturing key and participation constraints
- Functional dependencies
  - Attribute closure, Armstrong’s axioms
  - Determining candidate keys
  - Role in detecting data redundancy
- Schema refinement
  - Normalization
  - BCNF normalization process
- Capturing integrity constraints in relational schema
- General motivation and ideas from ORDBMS

Special Topics

- Distributed DBMS
  - Goals of data partitioning and data replication
    - Types of partitioning: range vs hash
  - Replication
    - Synchronous vs asynchronous
    - Strong vs. eventual/weak consistency
  - Challenges with distributed xacts (generally)
- NoSQL
  - CAP theorem
  - Query restrictions for performance (generally)
- Analytics
  - Generally what OLAP is, vs. OLTP, and what kinds of queries run