CS 133: Databases

Fall 2018
Lec 26 – 12/11
Data Analytics
Prof. Beth Trushkowsky

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)

Goals for Today

- Understand how Analytics processing (OLAP) is different than Transactional processing (OLTP)
- Reason about how data is organized and queried in a data warehouse
- Discuss current trends in Big Data 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

---

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, quarter, 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

Slicing the Cube

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

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

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;
```

Exercise 2 (a-c)

(a)

```sql
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)

```sql
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)

```sql
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;
```
Analysis: Cross-tabulation

- Sales from each dealer by car color
  - View popularized by spreadsheet applications

<table>
<thead>
<tr>
<th>Car color</th>
<th>Red</th>
<th>White</th>
<th>Blue</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dealer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>90K</td>
<td>30K</td>
<td>120K</td>
<td>240K</td>
</tr>
<tr>
<td>Bob</td>
<td>100K</td>
<td>10K</td>
<td>40K</td>
<td>150K</td>
</tr>
<tr>
<td>total</td>
<td>190K</td>
<td>40K</td>
<td>160K</td>
<td>390K</td>
</tr>
</tbody>
</table>

Also called “pivoting”

What about time dimension? “all” time values

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

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

The Data CUBE
Multidimensional OLAP (MOLAP)

- A CUBE relation: generalization of the cross-tabulation

| Darkest: aggregation over all three dimensions |
| Medium: aggregation over two dimensions |
| Lightest: aggregation over one dimension |

The CUBE Operator

- With k dimensions, there are $2^k$ possible SQL GROUP BY queries that can be generated

- Example: generic sales data
  - Measure Sales
  - Dimensions: Product (key: pid), Location (locid), Time (timeid)

  - CUBE pid, locid, timeid BY SUM Sales

  Child can be computed from any parent

Relationship of different groupings represented as a lattice
Materialized Views

- Computing GROUP BY and CUBE aggregates is expensive
  - OLAP queries perform these operations repeatedly

**Idea:** pre-compute and store the aggregates as *materialized views*

- Need to maintain as source data changes
  - Incremental updates?

View Selection

- Factors in deciding *which views to materialize*
  - What is its storage cost?
  - What is its update cost?
  - Which queries can benefit from it?
  - How much can a query benefit from it?

- E.g.,
  - Grouping by nothing: Small! Not as useful
  - Grouping all dimensions: useful, but large

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

Hadoop and HDFS

- HDFS – Hadoop File System

- HDFS properties:
  - Scalable to 1000s of nodes
  - Assume failures (hardware and software) are common
  - Targeted towards small numbers of very large files
  - Append only workloads: Write once, read multiple times
  - Traditional hierarchical file organization: directories and files

Open-source version of Google’s MapReduce

Thanks Mike Franklin
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

def mapper(line):
    for word in line.split():
        output(word, 1)

def reducer(key, values):
    output(key, sum(values))

Word Count Execution

<table>
<thead>
<tr>
<th>Input</th>
<th>Map</th>
<th>Shuffle &amp; Sort</th>
<th>Reduce</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>the quick brown fox</td>
<td>Map</td>
<td></td>
<td>Reduce</td>
<td>brown, 2</td>
</tr>
<tr>
<td>the fox</td>
<td>Map</td>
<td></td>
<td>Reduce</td>
<td>fox, 2</td>
</tr>
<tr>
<td>ate the mouse</td>
<td></td>
<td></td>
<td></td>
<td>how, 1</td>
</tr>
<tr>
<td>how now brown cow</td>
<td></td>
<td></td>
<td></td>
<td>now, 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the, 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

High Level Languages on Top

- HiveQL
- PigLatin
- jq script
- (High-level Languages)
- Hadoop MapReduce Dataflow Layer
- HBase Key-Value Store
- Hadoop Distributed File System
  (Byte-oriented file abstraction)
Pokémon or Big Data?

https://pixelastic.github.io/pokemonorbigdata/

Hadoop is Big Data!

Hadoop is a distributed system for counting words.

Next question