Scalable Data Analysis (CIS 602-02)

Data Cubes

Dr. David Koop
Polystores: BigDAWG

We first consider simple queries, ones which have comparable performance on all of an island's storage engines. We anticipate that many select-project-join queries will be in this category and we will examine in how to identify approaches such information. As a result, we propose a black box approach, then we will selectively add more sophisticated knowledge of individual systems, recognizing that this may make adding new storage engines to a polystore more challenging. More detailed knowledge might include the sizes of operands, their data distribution, available access methods, and explanations of query plans. If our query optimizer cannot be made robust using this approach, then we will selectively add more sophisticated knowledge of individual systems, recognizing that this may make adding new storage engines to a polystore more challenging.

Traditional query optimization is simply not capable of producing a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands. If he produces a different output, a user must specify the semantics he desires using SCOPE and CAST commands.
F1: OLTP and OLAP Together

- Distributed data storage: data is not stored at one central location
- Need to keep data and schemas in sync
- Hierarchical schemas keep data that is likely to be accessed at the same time together
- Optimistic Transactions: Long reads that keep track of timestamps and don't lock the database until the write happens
- Change History: Keep track of history as part of the database, also helps with caching
- DIY Object-Relational Mapping: don't automatically join or implicitly traverse relationships
- Protocol buffers as a way to store application data without translation + support for queries
Hierarchical Schema

Explicit table hierarchies. Example:

- **Customer** (root table): PK (CustomerId)
- **Campaign** (child): PK (CustomerId, CampaignId)
- **AdGroup** (child): PK (CustomerId, CampaignId, AdGroupId)

**Rows and PKs**

```
1
\|-- 1,3
\  |  \ 1,4
  |   \2,5
\  \  | 1,3,5
  |   \1,3,6
  |    \1,4,7
  |     \2,5,8
```

**Storage Layout**

- **Customer** (1)
- **Campaign** (1,3)
- **AdGroup** (1,3,5)
- **AdGroup** (1,3,6)
- **Campaign** (1,4)
- **AdGroup** (1,4,7)
- **Customer** (2)
- **Campaign** (2,5)
- **AdGroup** (2,5,8)

[Shute et al., 2012]
Clustered Storage

- Child rows under one root row form a **cluster**
- Cluster stored on one machine (unless huge)
- Transactions within one cluster are most efficient
- Very efficient joins inside clusters (can merge with no sorting)

**Rows and PKs**

```
  1
 / \
1,3 1,4
/ \ / \
1,3,5 1,3,6 1,4,7
```

**Storage Layout**

```
Customer (1)
Campaign (1,3)
AdGroup (1,3,5)
AdGroup (1,3,6)
Campaign (1,4)
AdGroup (1,4,7)
Customer (2)
Campaign (2,5)
AdGroup (2,5,8)
```

[Shute et al., 2012]
F1 Notes

• Schema changes: allow two different schemas at once
• Transaction types: Snapshot, Pessimistic, Optimistic
• Change History and application to caching
• Disk latency or network latency?
Project Proposal

• Due: Tuesday, October 27
• Types of projects
  - Dataset Analysis
  - Technique Improvement
• Requirements posted on myCourses
• 1-2 pages describing your goals
Midterm

- Main ideas of papers plus material covered during class
- Topics:
  - Python
  - Scientific Writing
  - Data Sources
  - Data Integration
  - Visualization
  - Statistics
  - Machine Learning
  - Clustering
  - Databases
  - Data Cubes
Midterm

- Format: Multiple Choice and Short Response
- Types of questions:
  - Difference between Frequentist and Bayesian analysis
  - Two cultures of statistical modeling
  - OLTP vs. OLAP
  - Types of operations on Data Cubes
  - Descriptive vs. Inferential Statistics
  - Mediated Integration vs. Data Warehouses
  - What are the advantages of dataspaces and pay-as-you-go integration?
  - Types of data sources
  - Structure of data (item, attribute, dataset types, data types)
Midterm

- What is the goal of a scientific research paper?
- Why is Python used for data analysis?
- What questions should you answer when critically reading a research paper?
- Why do scientific papers usually not cite blog posts?
- Describe differences between "big data" or "small data" problems.
- What makes data valuable?
- Why is data cleaning important? Give an example of cleaning data.
- What is a mixed-initiative interface? Give examples of systems that have such interfaces. Why are they useful?
- Why do we use data visualization when trying to understand datasets?
Midterm

- How does Voyager help users understand their data?

- Why does having a lot of data help smooth over the Frequentist vs. Bayesian debate?

- What are the two major categories of clustering algorithms? Which type is the BFR algorithm? How does BFR minimize execution time?

- Why have RDBMSs had few major architectural changes? Why have scientists started to question their utility in the past ten years?

- What is a polystore? How does this relate to federated databases?

- What is the purpose of data cubes?
OLTP vs. OLAP

• Online Transactional Processing (OLTP) often used in business applications, data entry and retrieval transactions

• OLTP Examples:
  - Add customer's shopping cart to the database of orders
  - Find me all information about John Hammond's death

• OLTP is focused on the day-to-day operations while Online Analytical Processing (OLAP) is focused on analyzing that data for trends, etc.

• OLAP Examples:
  - Find the average amount spent by each customer
  - Find which year had the most movies with scientists dying

• OLAP often done on read-only data warehouses with lots of historical data
Data Cubes

J. Han, M. Kamber, and J. Pei
Data Cube: A Lattice of Cuboids

- 0-D (apex) cuboid
- 1-D cuboids
- 2-D cuboids
- 3-D cuboids
- 4-D (base) cuboid

[Han et al., 2011]
Dimensions and Measures

• Dimensions: attributes than can be organized as a hierarchy
  - Parents summarize their children
  - Examples: time(day, week, month, quarter, year), location(country, region, state, city, district)

• Measures: A number
  - Examples: number of orders, forecasted revenue
  - Types:
    • Distributive: When result applied to n aggregate values equals the result applied to all values (e.g. count, sum, min, max)
    • Algebraic: When result can be computed by a function on distributive measures (e.g. mean, stddev)
    • Holistic: When no constant bound on storage size needed to store a subaggregation (e.g. median, mode, rank)
Cube Operations

• Roll-up: aggregate up the given hierarchy
• Drill-down: refine down the given hierarchy
• Roll-up and drill-down are "inverses"
Nanocubes for Real-Time Exploration of Spatiotemporal Datasets

L. Lins, J. T. Klosowski, and C. Scheidegger
Data Cube Aggregations

<table>
<thead>
<tr>
<th>Relation A</th>
<th>Cube on Device, Language D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
<td><strong>Device</strong></td>
</tr>
<tr>
<td>US</td>
<td>Android</td>
</tr>
<tr>
<td>US</td>
<td>iPhone</td>
</tr>
<tr>
<td>South Africa</td>
<td>iPhone</td>
</tr>
<tr>
<td>India</td>
<td>Android</td>
</tr>
<tr>
<td>Australia</td>
<td>iPhone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aggregation B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
</tr>
<tr>
<td>All</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group By on Device, Language C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
</tr>
<tr>
<td>All</td>
</tr>
<tr>
<td>All</td>
</tr>
<tr>
<td>All</td>
</tr>
</tbody>
</table>

Equivalent to Group By on all possible subsets of \{Device, Language\} [Lins et. al, 2013]
Nanocube Queries

- Representing natural language queries as data cube queries

<table>
<thead>
<tr>
<th>Natural language query</th>
<th>s</th>
<th>c</th>
<th>t</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>count of all Delta flights</td>
<td>R</td>
<td>U</td>
<td>R</td>
<td>U /where/carry=Delta</td>
</tr>
<tr>
<td>count of all Delta flights in the Midwest</td>
<td>R</td>
<td>Midwest</td>
<td>R</td>
<td>U /region/Midwest/where/carry=Delta</td>
</tr>
<tr>
<td>count of all flights in 2010</td>
<td>R</td>
<td>U</td>
<td>D</td>
<td>2010 /field/carry/when/2010</td>
</tr>
<tr>
<td>heatmap of Delta flights in 2010</td>
<td>D</td>
<td>tile0</td>
<td>R</td>
<td>2010 /tile/tile0/when/2010/where/carry=Delta</td>
</tr>
</tbody>
</table>

- s = space, c = category, t = time
- R = rollup, D = drill down
- <value> after RD = subset of dimension's domain, U = universe
- Note that time queries are stored in an array of cumulative counts

[Lins et. al, 2013]
Building a Nanocube

Indexing Schema

\[ S = [\ell_{\text{spatial1}}, \ell_{\text{spatial2}}, \ell_{\text{device}}] \]

Five Tweets: Location and Device

- Device: \( \ell_{\text{device}}( \text{O} ) = \text{Android} \)
- \( \ell_{\text{device}}( \bullet ) = \text{iPhone} \)

1. \( \ell_{\text{spatial1}} \)

2. \( \ell_{\text{spatial2}} \)

3. Parent-child (same dimension):

4. Content (next dimension):

5. Updated in current step

Dimension boundary

[Lins et. al, 2013]
**Building a Nanocube: Step 1**

![Five Tweets: Location and Device](image)

\[ \ell_{\text{device}}(\circ) = \text{Android} \]
\[ \ell_{\text{device}}(\bullet) = \text{iPhone} \]

**Indexing Schema**

\[ S = [ [\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}] ] \]

---

parent-child (same dimension):

proper \hspace{1cm} shared

content (next dimension):

proper \hspace{1cm} shared

updated in current step

dimension boundary

---

[Lins et. al, 2013]
Building a Nanocube: Step 2

Five Tweets: Location and Device

\[ \ell_{\text{device}}(\bigcirc) = \text{Android} \]
\[ \ell_{\text{device}}(\bullet) = \text{iPhone} \]

Indexing Schema
\[ S = [\ell_{\text{spatial1}}, \ell_{\text{spatial2}}, \ell_{\text{device}}] \]

parent-child (same dimension):
- proper
- shared

content (next dimension):
- proper
- shared

updated in current step
dimension boundary

[Becker et al., 2001; Ryzhikov et al., 2013; Mackinlay's Automatic]

Mackinlay's Automatic
Building a Nanocube: Step 3

3.

parent-child (same dimension):
- proper
- shared

content (next dimension):
- proper
- shared

updated in current step
- dimension boundary

Five Tweets: Location and Device

\( \ell_{\text{device}}(O) = \text{Android} \)
\( \ell_{\text{device}}(\bullet) = \text{iPhone} \)

Indexing Schema
\[
S = [\ell_{\text{spatial1}} \cdot \ell_{\text{spatial2}}] \cdot \ell_{\text{device}}
\]

[Lins et. al, 2013]
Building a Nanocube: Step 4

By the same token, nanocubes support temporal queries at multiple scales, such as for a single thread running on computers ranging from laptops, to iPhone 6, or on the front-end [Lins et. al, 2013]. In contrast, some of the work in large data visualization involves expansions along the wanted dimensions. Nanocubes also provide relational structure of the data with the graphical primitives available almost since the field’s inception [Lins et. al, 2013].

As we will show, nanocubes lend themselves very well to building multidimensional, spatiotemporal datasets; and these visualizations at a variety of dataset sizes and scales.

Statistical sampling can be performed on the database backend [Image 176x3 to 266x355] that can be updated in current step dimension boundary.

In addition, as Liu et al. argue, sampling by itself is not preferable. In addition, as Liu et al. argue, sampling by itself is not necessary because the speed of our data cube structure is operating, the result is indistinguishable (or close to) from a complete large dataset will still be too slow for fluid interaction, even with GPUs.

Seeking Mantra [Image 29x313 to 313x369] that can provide insights from the data. In additional to the visualizations and overviews, filters, zooming, and details-on-demand inside the spa-

D. Koop, CIS 602-02, Fall 2015

24
Building a Nanocube: Step 5

5.

parent-child (same dimension):

proper

shared

content (next dimension):

proper

shared

updated in current step

dimension boundary

Five Tweets: Location and Device

Indexing Schema

\[ S = \left[ \ell_{\text{spatial1}}, \ell_{\text{spatial2}}, \ell_{\text{device}} \right] \]

[Lins et. al, 2013]
Nanocubes Discussion

• Save space by organizing the data in a manner that takes advantage of data sparseness
• Limited to one spatial dimension, one temporal dimension
Next Week

• No scheduled office hours, will be available via email
• October 27: **No class**, Project Proposal Due
• October 29: Midterm Exam
• Next: Cloud/Cluster Computing + Assorted Topics