Scalable Data Analysis (CIS 602-02)

Data Cubes

Dr. David Koop
Polystores: BigDAWG

As a result, the BigDAWG query language consists of multiple scopes to indicate the expected behavior of operands, their data distribution, available access methods, and explanations of query plans. We anticipate that many select-project-join queries will be comparable performance on all of an island's storage engines. If the signature of a given cross-island operation does not require any data movement, at the end of Stage 1, we are left with computations on collections of operands, their data distribution, available access methods.

Simple Queries

For such queries we propose to minimize data conversions and unnecessary network traffic. Rather, the optimizer would have to adapt whenever a new storage engine is added to the polystore. Lastly, conventional optimizers 

[19] require the planner to maintain a model for supporting cross-database queries. First, cost-based optimizers 

[19] require the planner to multi-island query planning.

Second, the optimizer to multi-island query planning.

We first consider simple queries, ones which have com-
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
   /     /   /
1,3,5 1,3,6 1,4,7 2,5
```

### 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

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 that 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 subaggregate (e.g. median, mode, rank) [Han et al., 2011]
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</th>
<th>A</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</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
<td><strong>Device</strong></td>
</tr>
<tr>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Cube on <strong>Device, Language</strong></th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
<td><strong>Device</strong></td>
</tr>
<tr>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>All</td>
<td>Android</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
</tr>
<tr>
<td>All</td>
<td>Android</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</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/carrier=Delta</td>
</tr>
<tr>
<td>count of all Delta flights in the Midwest</td>
<td>R Midwest</td>
<td>R</td>
<td>R</td>
<td>U /region/Midwest/where/carrier=Delta</td>
</tr>
<tr>
<td>count of all flights in 2010</td>
<td>R</td>
<td>U</td>
<td>D</td>
<td>2010 /field/carrier/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/carrier=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

As we will show, nanocubes lend themselves very well to building exploratory visualization on datasets that are multidimensional, spatiotemporal datasets. By Careful data aggregation [Lins et. al, 2013], or on the front-end [Show Me], nanocubes enable scalable solutions for low-latency large data graphics. While Elmqvist et al. [2000] presents data cube construction techniques necessary for display [ORK 2000], the enormous power of graphics processing units has also become necessary for shipping the computation and data to a cluster of processing nodes. For display [ORK 2000], does not necessarily help achieve low latency, which is essential for interactive response times, each entry can have additional attributes (see section 3). Spatial sampling can be performed on the database backend [Lins et. al, 2013], but without algorithmic changes, linear scans through the dataset will still be too slow for fluid interaction, even with GPUs. Techniques such as query prediction become necessary [Elmqvist et al, 2000], which is essential for real-time aggregation as a first-class citizen [Sismanis et al, 2011]. These techniques, together with experiments to measure its utilization of spatial databases, such as counting events in a spatial region that can be either a rectangle covering most of the world, or a heatmap (next dimension). Another popular way to cope with large datasets is through sampling. Statistical sampling can be performed on the database backend [Lins et. al, 2013], or on the front-end [Show Me].

Indexing Schema

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

Five Tweets: Location and Device

- 01,00
- 01,01
- 01,10
- 10,01
- 10,10
- 10,11

0,0 0,1 1,0 1,1

\[ l_{\text{device}}(\bigodot) = \text{Android} \]

\[ l_{\text{device}}(\bullet) = \text{iPhone} \]
Building a Nanocube: Step 1

1. 

parent-child (same dimension):

content (next dimension):

updated in current step

dimension boundary

Five Tweets: Location and Device

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

Indexing Schema

\[ S = \begin{pmatrix} l_{\text{spatial1}}, l_{\text{spatial2}} \end{pmatrix} \]

[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

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

[\text{Lins et. al, 2013}]
Building a Nanocube: Step 3

3.

iPhone

Android

- \(O_1\)
- \(O_2\)
- \(O_3\)

- \(O_1\)
- \(O_2\)
- \(O_3\)

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

content (next dimension):
- proper
- shared

Updated in current step

Dimension boundary

D. Koop, CIS 602-02, Fall 2015

[UMass | Dartmouth]
Building a Nanocube: Step 4

4.

parent-child (same dimension):
proper
shared

content (next dimension):
proper
shared

updated in current step
dimension boundary

Indexing Schema

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

Five Tweets: Location and Device

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

D. Koop, CIS 602-02, Fall 2015
Building a Nanocube: Step 5

5.

parent-child (same dimension):

proper

shared

content (next dimension):

proper

shared

updated in current step
dimension boundary

Indexing Schema

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

[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