INCREMENTAL, APPROXIMATE DATABASE QUERIES AND UNCERTAINTY FOR EXPLORATORY VISUALIZATION

Danyel Fisher
Microsoft Research
Exploratory Visualization

Initial Query

Process query

Get a response

Change parameters
Handling Big Data for Infovis

• Megabytes: More data than there are pixels on screen
  – Need to summarize, zoom

• Gigabytes: More bits than there are in memory
  – Need to think

• Terabytes: More bits than there are on a single disk
  – Yo

Extreme Visualization:
Squeezing a Billion Records into a Million Pixels

Ben Shneiderman
Human-Computer Interaction Lab & Department of Computer Science
University of Maryland
College Park, MD 20742
ben@cs.umd.edu
Big Data Visualization

Initial Query

Process query for a while

Get a response

Change parameters
Exploratory Big Data Visualization

Initial Query → Process query → Get a good enough, quick response → Change parameters → Initial Query
Happiness over time

Time

100%

Online

Traditional
What is “good enough”?

• “I can act on this query”
• “I realize that this query is incorrect”
  – create a new query
• “I want a detailed response”
  – Wait for the full query to complete
What is “quick”? 

- Milliseconds: Feels real-time 
- Seconds: Laggy but possible 
- Minutes: Forget context 
- Hours: Forget question
Part I: INCREMENTAL DATABASES
Part II: VISUALIZATIONS
Part III: A PROTOTYPE
Part I

INCREMENTAL DATABASES
Techniques for Speeding Big Data

• **Pre-aggregate (e.g. OLAP)**
  – Fast but inflexible

• **Parallel Computation**
  – Hadoop Pig, Sawzall, DryadLinq: use existing data structures and add visualizations
  – Dremel: Use novel data structures

• **Sampling & approximate queries**
Control (1999)

<table>
<thead>
<tr>
<th>College</th>
<th>Enrolled</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>190</td>
<td>2.9353933</td>
</tr>
<tr>
<td>B</td>
<td>208</td>
<td>3.0174129</td>
</tr>
<tr>
<td>D</td>
<td>186</td>
<td>3.5308642</td>
</tr>
<tr>
<td>E</td>
<td>167</td>
<td>3.0384614</td>
</tr>
<tr>
<td>H</td>
<td>177</td>
<td>3.1497006</td>
</tr>
<tr>
<td>K</td>
<td>191</td>
<td>3.0500000</td>
</tr>
<tr>
<td>L</td>
<td>185</td>
<td>3.0730338</td>
</tr>
<tr>
<td>M</td>
<td>209</td>
<td>3.1683416</td>
</tr>
<tr>
<td>N</td>
<td>175</td>
<td>3.2116563</td>
</tr>
<tr>
<td>P</td>
<td>177</td>
<td>3.1225805</td>
</tr>
<tr>
<td>Q</td>
<td>169</td>
<td>3.3609271</td>
</tr>
<tr>
<td>R</td>
<td>184</td>
<td>3.5167599</td>
</tr>
<tr>
<td>V</td>
<td>183</td>
<td>3.2289157</td>
</tr>
<tr>
<td>Z</td>
<td>22</td>
<td>3.0227273</td>
</tr>
</tbody>
</table>

![GPA per College Graph]

- High GPA
- Low GPA
Sampling and Approximate Queries

• Joe Hellerstein (et al)’s CONTROL project
• General concept:
  – Grab a little bit (more) of the database quickly
  – Estimate value & size of confidence interval
  – Repeat
• What can we do? **Aggregate.**
  – Some aggregates are very good. **AVERAGE. COUNT. SUM.**
  – Some aggregates are really bad. **MAX (or Top-K). MIN.**
  – Some aggregates have loose approximations. **PERCENTILE. COUNT DISTINCT.**
Is that powerful enough?

• Some things are just histograms:
  – Bar chart (sum)
  – Tag Clouds
  – Treemap (multilevel sum)

• Don’t do a scatterplot, do a 2D histogram

• Even some machine learning:
  – K-Means: Locate average of group, find centroids, repeat
Computing Confidence Interval

• Estimator
  – Total elements
  – Mean seen so far
  – Number of elements that cross the filter so far

• Intuition: if you know about how the data you’ve seen so far behaves, you can guess the rest.

Based on:
  – Standard deviation seen so far
  – Also nice: data min/max
  – Some theorems: std dev overall
Why Not Just Do a Straight Sample?

• Don’t know how good you are without confidence intervals
• May need *larger* sample (over memory) to get tight intervals.
• How big is big enough?
Why Isn’t Everyone Doing This?

• A good sample is random ... but a random sample requires accessing (potentially) all rows
• Need to maintain some data for bounds
  – E.g. column min, max
• Databases don’t support incremental callbacks
• Joins can be tricky (but NoSQL?)
Part II

THE VISUALIZATION CHALLENGES
Uncertainty Visualization

• “Confidence” is something like “uncertainty”
• Lots of sources of uncertainty have been studied
  – Credibility of sources
  – Model uncertainty
  – Simulation uncertainty
  – Incompleteness

• *Statistical and Quantitative Uncertainty*
Figure 1: Error bars and ambiguation applied to some common chart types.
Streit & Pham (2008)

Fig. 1. Visualizations of employment numbers in California. Years 2005-2010 are predicted. (a) Assuming the average growth, (b) Indicating the likely range of employment numbers, (c) The area with the highest likelihood indicates the likely range. (d) The area with the highest probability indicates the area of likely employment numbers.
User Study of Uncertainty
Sanyal Zhang et al (2009)

No single technique was clear winner
Part III

PROTOTYPING IT
Why Prototype a Front End?

• Back-End
  – Technical implementation
  – Adapt to NoSQL
  – Ways to guide sampling

• Front-End
  – What visual codings work beside error bars? How do we extend to multiple dimensions?
  – How good is “good enough”?
    How fast is “fast enough”?
  – What sorts of problems work well?
  – User experience of these systems
  – Reducing communication overhead
A Full System View

- **Terabyte Server or Distributed System**
- **Query**
- **Data query**
- **Compute**
- **Front End**
- **Aggregate Value Metadata**
- **Estimator Confidence interval**
The Desktop Edition

Query

Little Tiny Database + Compute

Front End

Estimator
Error range
Call to Action

There’s work to be done here and a very compelling source of approximate data.

Let’s build these!
Thank you!
(this work is not sponsored by DOE)

DANYELF@MICROSOFT.COM
RESEARCH.MICROSOFT.COM/~DANYELF
Joins

• Lots of database research dedicated to joins
  – “Hash ripple” join
  – 10% sample, twice, is a 1% sample assuming independence

• Sentinel joins:
  – Some joins are impossible to do incrementally
    E.g. (select count(direct reports) where manager=president)

• Or live without them
  • NoSQL has very limited (and expensive) joining
  • Denormalized tables for distributed computation