A Quick Introduction to Approximate Query Processing

CS286, Spring'2007
Minos Garofalakis
Outline

- Intro & Approximate Query Answering Overview
  - Synopses, System architectures, Commercial offerings
- One-Dimensional Synopses
  - Histograms, Samples, Wavelets
- Multi-Dimensional Synopses and Joins
  - Multi-D Histograms, Join synopses, Wavelets
- Set-Valued Queries
  - Using Histograms, Samples, Wavelets
- Discussion & Comparisons
- Advanced Techniques & Future Directions
  - Dependency-based, Workload-tuned, Streaming data
Decision Support Systems

- **Data Warehousing**: Consolidate data from many sources in one large repository.
  - Loading, periodic synchronization of replicas.
  - Semantic integration.

- **OLAP**:
  - Complex SQL queries and views.
  - Queries based on spreadsheet-style operations and “multidimensional” view of data.
  - Interactive and “online” queries.

- **Data Mining**:
  - Exploratory search for interesting trends and anomalies. (Another lecture!)
Introduction & Motivation

Decision Support Systems (DSS) → SQL Query → Exact Answer → Long Response Times!

• Exact answers **NOT** always required
  - DSS applications usually *exploratory*: early feedback to help identify “interesting” regions
  - *Aggregate queries*: precision to “last decimal” not needed
    • e.g., “What percentage of the US sales are in NJ?” (display as bar graph)
  - *Preview* answers while waiting. *Trial* queries
  - Base data can be *remote or unavailable*: approximate processing using locally-cached **data synopses** is the only option
Fast Approximate Answers

- Primarily for Aggregate Queries
- Goal is to quickly report the leading digits of answers
  - In seconds instead of minutes or hours
  - Most useful if can provide error guarantees

E.g., Average salary

\$59,000 +/- \$500 \text{(with 95\% confidence)} \text{ in 10 seconds}
\text{vs.} \ \$59,152.25 \text{ in 10 minutes}

- Achieved by answering the query based on samples or other synopses of the data
- Speed-up obtained because synopses are \textit{orders of magnitude smaller} than the original data
Approximate Query Answering

Basic Approach 1: Online Query Processing

- e.g., Control Project [HHW97, HH99, HAR00]
- Sampling at query time
- Answers continually improve, under user control
Approximate Query Answering

Basic Approach 2: Precomputed Synopses
- Construct & store synopses prior to query time
- At query time, use synopses to answer the query

- Like estimation in query optimizers, but
  • reported to the user (need higher accuracy)
  • more general queries

- Need to maintain synopses up-to-date

- Most work in the area based on the precomputed approach
  • e.g., Sample Views [OR92, Olk93], Aqua Project [GMP97a, AGP99, etc]
The Aqua Architecture

Picture without Aqua:

- User poses a query Q
- Data Warehouse executes Q and returns result
- Warehouse is periodically updated with new data
Picture with Aqua:

- Aqua is middleware, between the user and the warehouse
- Aqua Synopses are stored in the warehouse
- Aqua intercepts the user query and rewrites it to be a query $Q'$ on the synopses. Data warehouse returns approximate answer
Online vs. Precomputed

Online:

+ **Continuous refinement** of answers (online aggregation)
+ **User control:** what to refine, when to stop
+ **Seeing the query** is very helpful for fast approximate results
+ **No maintenance overheads**
+ See [HH01] Online Query Processing tutorial for details

Precomputed:

+ **Seeing entire data** is very helpful (provably & in practice)
  (But must construct synopses for a **family** of queries)
+ **Often faster:** better access patterns,
  small synopses can reside in **memory** or cache
+ **Middleware:** Can use with any DBMS, no special index striding
+ **Also effective** for **remote** or **streaming** data
Commercial DBMS

- **Oracle, IBM Informix:** Sampling operator (online)

- **IBM DB2:** “IBM Almaden is working on a prototype version of DB2 that supports sampling. The user specifies a priori the amount of sampling to be done.”

- **Microsoft SQL Server:** “New auto statistics extract statistics [e.g., histograms] using fast sampling, enabling the Query Optimizer to use the latest information.” The index tuning wizard uses sampling to build statistics.
  - see [CN97, CMN98, CN98]

  **In summary, not much announced yet**
Approximate Query Processing using Data Synopses

Decision Support Systems (DSS)

Compact Data Synopses

SQL Query

Exact Answer

Long Response Times!

“Transformed” Query

Approximate Answer

FAST!!

• How to construct effective data synopses??
Outline

• Intro & Approximate Query Answering Overview

• One-Dimensional Synopses
  - Histograms: Equi-depth, Compressed, V-optimal, Incremental maintenance, Self-tuning
  - Samples: Basics, Sampling from DBs, Reservoir Sampling
  - Wavelets: 1-D Haar-wavelet histogram construction & maintenance

• Multi-Dimensional Synopses and Joins

• Set-Valued Queries

• Discussion & Comparisons

• Advanced Techniques & Future Directions
Relations as Frequency Distributions

One-dimensional distribution

<table>
<thead>
<tr>
<th>Age</th>
<th>Tuple Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>35</td>
<td>20</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
</tr>
</tbody>
</table>

Three-dimensional distribution

<table>
<thead>
<tr>
<th>Age</th>
<th>Sales</th>
<th>Tuple Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>35</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Salary</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>MG</td>
<td>34</td>
<td>100K</td>
<td>25K</td>
</tr>
<tr>
<td>JG</td>
<td>33</td>
<td>90K</td>
<td>30K</td>
</tr>
<tr>
<td>RR</td>
<td>40</td>
<td>190K</td>
<td>55K</td>
</tr>
<tr>
<td>JH</td>
<td>36</td>
<td>110K</td>
<td>45K</td>
</tr>
<tr>
<td>MF</td>
<td>39</td>
<td>150K</td>
<td>50K</td>
</tr>
<tr>
<td>DD</td>
<td>45</td>
<td>150K</td>
<td>50K</td>
</tr>
<tr>
<td>JN</td>
<td>43</td>
<td>140K</td>
<td>45K</td>
</tr>
<tr>
<td>AP</td>
<td>32</td>
<td>70K</td>
<td>20K</td>
</tr>
<tr>
<td>EM</td>
<td>24</td>
<td>50K</td>
<td>18K</td>
</tr>
<tr>
<td>DW</td>
<td>24</td>
<td>50K</td>
<td>28K</td>
</tr>
</tbody>
</table>
Histograms

- Partition attribute value(s) domain into a set of buckets

- Issues:
  - How to partition
  - What to store for each bucket
  - How to estimate an answer using the histogram

- Long history of use for selectivity estimation within a query optimizer [Koo80], [PSC84], etc.

- [PIH96] [Poo97] introduced a taxonomy, algorithms, etc.
1-D Histograms: Equi-Depth

- **Goal**: Equal number of rows per bucket (B buckets in all)

- **Can construct** by first sorting then taking B-1 equally-spaced splits

- **Faster construction**: Sample & take equally-spaced splits in sample
  - Nearly equal buckets
  - Can also use one-pass quantile algorithms (e.g., [GK01])
1-D Histograms: Equi-Depth

- Can **maintain** using one-pass algorithms (insertions only), or

- Use a backing sample [GMP97b]: Maintain a larger sample on disk in support of histogram maintenance
  - Keep histogram **bucket counts** up-to-date by incrementing on row insertion, decrementing on row deletion
  - **Merge** adjacent buckets with small counts
  - **Split** any bucket with a large count, using the sample to select a split value, i.e., take median of the sample points in bucket range
    - Keeps counts within a factor of 2; for more equal buckets, can recompute from the sample
1-D Histograms: Compressed

- Create singleton buckets for largest values, equi-depth over the rest

- Improvement over equi-depth since get exact info on largest values, e.g., join estimation in DB2 compares largest values in the relations

**Construction:** Sorting + O(B log B) + one pass; can use sample

**Maintenance:** Split & Merge approach as with equi-depth, but must also decide when to create and remove singleton buckets [GMP97b]
1-D Histograms: V-Optimal

[IP95] defined V-optimal & showed it minimizes the average selectivity estimation error for equality-joins & selections
- Idea: Select buckets to minimize frequency variance within buckets

- [JKM98] gave an O(B*N^2) time dynamic programming algorithm
  - F[k] = freq. of value k; AVGF[i:j] = avg freq for values i..j
  - SSE[i:j] = sum{k=i..j}F[k]^2 - (j-i+1)*AVGF[i:j]^2
  - For i=1..N, compute P[i] = sum{k=1..i} F[k] & Q[i] = sum{k=1..i} F[k]^2
  - Then can compute any SSE[i:j] in constant time
  - Let SSEP(i,k) = min SSE for F[1]..F[i] using k buckets
  - Then SSEP(i,k) = min{j=1..i-1} (SSEP(j,k-1) + SSE[j+1:i]), i.e., suffices to consider all possible left boundaries for kth bucket
  - Also gave faster approximation algorithms
Answering Queries: Equi-Depth

Answering queries:

- select count(*) from R where 4 <= R.A <= 15
- approximate answer: F * |R|/B, where
  - F = number of buckets, including fractions, that overlap the range
  - error guarantee: ± 2 * |R|/B

answer: 3.5 * |R|/6 ± 0.5 * |R|/6
Answering Queries: Histograms

- Answering queries from 1-D histograms (in general):
  - (Implicitly) map the histogram back to an approximate relation, & apply the query to the approximate relation

- Continuous value mapping [SAC79]:

- Uniform spread mapping [PIH96]:

  Count spread evenly among bucket values

  Need number of distinct in each bucket

CS286, Spring’07 – Minos Garofalakis  #21
1. Tune Bucket Frequencies:

- Compare actual selectivity to histogram estimate
- Use to adjust bucket frequencies

Actual = 60
Estimate = 40
Error = +20

- Divide d*Error proportionately, d=dampening factor

d=\frac{1}{2} \text{ of Error} = +10
So divide +4,+3,+3
2. Restructure:

- Merge buckets of near-equal frequencies
- Split large frequency buckets

Also Extends to Multi-D
Sampling: Basics

Idea: A small random sample $S$ of the data often well-represents all the data

- For a fast approx answer, apply the query to $S$ & “scale” the result
- E.g., $R.a$ is $\{0,1\}$, $S$ is a 20% sample
  - select count(*) from $R$ where $R.a = 0$
  - select $5 \times$ count(*) from $S$ where $S.a = 0$
  - Est. count = $5 \times 2 = 10$, Exact count = 10

Unbiased: For expressions involving count, sum, avg: the estimator is unbiased, i.e., the expected value of the answer is the actual answer, even for (most) queries with predicates!

- Leverage extensive literature on confidence intervals for sampling
  - Actual answer is within the interval $[a,b]$ with a given probability
  - E.g., $54,000 \pm 600$ with prob $\geq 90\%$
**Sampling: Confidence Intervals**

<table>
<thead>
<tr>
<th>Method</th>
<th>90% Confidence Interval (±)</th>
<th>Guarantees?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Limit Theorem</td>
<td>$1.65 \times \sigma(S) / \sqrt{</td>
<td>S</td>
</tr>
<tr>
<td>Hoeffding</td>
<td>$1.22 \times (\text{MAX-MIN}) / \sqrt{</td>
<td>S</td>
</tr>
<tr>
<td>Chebychev (known $\sigma(R)$)</td>
<td>$3.16 \times \sigma(R) / \sqrt{</td>
<td>S</td>
</tr>
<tr>
<td>Chebychev (est. $\sigma(R)$)</td>
<td>$3.16 \times \sigma(S) / \sqrt{</td>
<td>S</td>
</tr>
</tbody>
</table>

**Confidence intervals for Average:** select avg(R.A) from R
(Can replace R.A with any arithmetic expression on the attributes in R)
$\sigma(R) = \text{standard deviation of the values of R.A;} \quad \sigma(S) = \text{s.d. for S.A}$

- If predicates, S above is subset of sample that satisfies the predicate
- Quality of the estimate depends only on the **variance in R & |S| after the predicate:** So 10K sample may suffice for 10B row relation!
  - Advantage of larger samples: can handle more selective predicates
Sampling from Databases

- Sampling disk-resident data is slow
  - Row-level sampling has high I/O cost:
    - must bring in entire disk block to get the row
  - Block-level sampling: rows may be highly correlated
  - Random access pattern, possibly via an index
  - Need acceptance/rejection sampling to account for the variable number of rows in a page, children in an index node, etc

- Alternatives
  - Random physical clustering: destroys “natural” clustering
  - Precomputed samples: must incrementally maintain (at specified size)
    - Fast to use: packed in disk blocks, can sequentially scan, can store as relation and leverage full DBMS query support, can store in main memory
One-Pass Uniform Sampling

- Best choice for incremental maintenance
  - Low overheads, no random data access

- Reservoir Sampling [Vit85]: Maintains a sample $S$ of a fixed-size $M$
  - Add each new item to $S$ with probability $M/N$, where $N$ is the current number of data items
  - If add an item, evict a random item from $S$
  - Instead of flipping a coin for each item, determine the number of items to skip before the next to be added to $S$

- To handle deletions, permit $|S|$ to drop to $L < M$, e.g., $L = M/2$
  - remove from $S$ if deleted item is in $S$, else ignore
  - If $|S| = M/2$, get a new $S$ using another pass (happens only if delete roughly half the items & cost is fully amortized) [GMP97b]
Biased Sampling

- Often, advantageous to sample different data at different rates (Stratified Sampling)
  - E.g., outliers can be sampled at a higher rate to ensure they are accounted for; better accuracy for small groups in group-by queries
  - Each tuple j in the relation is selected for the sample S with some probability Pj (can depend on values in tuple j)
  - If selected, it is added to S along with its scale factor $sf = 1/Pj$

- Answering queries from S: e.g.,
  - select $\text{sum}(R.a)$ from R where $R.b < 5$
  - select $\text{sum}(S.a \times S.sf)$ from S where $S.b < 5$

  - Unbiased answer. Good choice for Pj's results in tighter confidence intervals

\[
\begin{array}{ccccccc}
\text{R.a} & 10 & 10 & 10 & 50 & 50 \\
\text{Pj} & 1/3 & 1/3 & 1/3 & 1/2 & 1/2 \\
\text{S.sf} & -- & 3 & -- & -- & 2 \\
\text{Sum}(R.a) = 130 \\
\text{Sum}(S.a*S.sf) = 10*3 + 50*2 = 130
\end{array}
\]