Survey of Approximate Query Processing Techniques

Gautam Das
University of Texas at Arlington
http://ranger.uta.edu/~gdas/

Acknowledgements
- Brian Babcock (Stanford)
- Kaushik Chakrabarti (MSR)
- Mayur Datar (Stanford)
- Minos Garofalakis (Bell Labs)
- Venky Ganti (MSR)

Outline
- Introduction
- Non-Sampling Based Methods
  - Histograms
  - Wavelets
- Sampling Based Methods
  - Uniform Random Sampling
  - Biased/Non-Uniform Random Sampling
  - Distinct Value Estimation
- Conclusions

Introduction
- Approximate Query Processing (AQP) in Relational Databases
- Focus on answering Aggregate queries
  - Aggregation queries often arise in OLAP and Data-Mining for large Databases

Motivation for AQP
- Answering complex aggregation queries over large data warehouses exactly can be time consuming
- If approximate answers are acceptable, can we answer these queries much faster?
  - Along with approximate answers, an estimate of the error in the approximation is important
    - Confidence Intervals

Examples of Aggregate Queries
- SPJ queries with
  - aggregation operators such as
    - Count, Sum, Count(Distinct), Avg
  - Group By

Examples of Aggregate Queries
- SPJ queries with
  - aggregation operators such as
    - Count, Sum, Count(Distinct), Avg
  - Group By
Example

```sql
SELECT State, COUNT(*) as ItemCount
FROM SalesData
WHERE ProductID = 5437
GROUP BY State
ORDER BY ItemCount DESC
```

<table>
<thead>
<tr>
<th>State</th>
<th>Exact Answer</th>
<th>Approximate Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>3425</td>
<td>WA 3400 50</td>
</tr>
<tr>
<td>CA</td>
<td>1065</td>
<td>CA 1150 100</td>
</tr>
<tr>
<td>WY</td>
<td>432</td>
<td>WY 400 50</td>
</tr>
<tr>
<td>TN</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>KY</td>
<td>56</td>
<td></td>
</tr>
</tbody>
</table>

Estimates may have errors
Small groups may be missing

Quantifying Error

- Consider a simple aggregation query
  - Answer is a single number
  - Relative error of query Q
    \[ Error(Q) = \frac{\text{ExactAnswer} - \text{ApproxAnswer}}{\text{ExactAnswer}} \]
  - For GROUP-BY query
    - Error of query = average error per group

Basic Approach

- Synopsis of data
  - Computed online
  - Pre-computed
- Rewrite query to run against synopsis
- Give error estimate

Different Synopses

- Histograms, Wavelets, Samples
- Comparing different methods
  - Changes necessary to the SQL engine
  - Accuracy and confidence intervals
  - Resilience to high dimensional data
  - Maintenance over updates
    - not the focus of this talk

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

1-D Histograms: Equi-Depth

- Goal: Equal number of rows per bucket
  - 8 buckets in all
**Equi-Depth Histogram Construction**
- First sort, 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

**Answering Queries**
- Select count(*) from R where 4 <= R.A <= 15
- Approximate answer: \( F \times |R| / B \), where
  - \( F \) = number of buckets, including fractions, that overlap the range
  - Error guarantee: \( \pm 2 \times |R| / B \)
  - Answer: \( 3.5 \times |R| / B + 0.5 \times |R| / B \)

**Multi-Dimensional Histograms**
- Use small number of multi-dimensional buckets to directly approximate the joint data distribution
- Uniform spread & frequency approximation within buckets

**1-D Histograms: V-Optimal**
- Select buckets to minimize frequency variance within buckets
  - [Ioannidis, Poosala 95]
- \( O(B^k \times N^2) \) time dynamic programming algorithm [Jagadish et al, 98]

**Multi-Dimensional Histogram Construction**
- Construction problem is much harder even for two dimensions
  - [Muthukrishnan et al 99]
- Multi-dimensional equi-depth histograms
  - Fix an ordering of the dimensions \( A_1, A_2, \ldots, A_k \)
  - \( a = k^{th} \) root of desired no. of buckets
  - Initialize \( B = \) data distribution
  - For \( i = 1, \ldots, k \)
    - Split each bucket in \( B \) into an equi-depth partition along \( A_i \)
    - Return resulting buckets
- Problems: limited set of bucketizations; fixed \( a \) and fixed dimension ordering can result in poor partitioning
Multi-Dimensional Histogram Construction

- **MHIST histograms** [Poosala, Ioannidis 97]
  - At each step
    - Choose the bucket \( b \) in \( B \) containing the attribute \( A_i \) whose marginal is the most in need of partitioning
    - Split \( b \) along \( A_i \) into \( p \) (e.g., \( p=2 \)) buckets

- **MHIST** chooses bucket/dimension to split based on its criticality, thus allows for much larger class of bucketizations (hierarchical space partitioning)
- Experimental results verify superiority over equi-depth

Other Multi-Dimensional histograms

- **GenHist** [Gunopulos et al 90]
  - Allows for overlapping buckets
- **STHoles** [Bruno et al 01]
  - Workload based
  - Considers nested buckets

Histograms: Summary

- 1-Dimensional histograms very useful in selectivity estimation
- Multi-dimensional histograms suffer from the curse of dimensionality
  - Model of the joint distribution becomes inaccurate as dimensions increase
- Histograms as AQP systems require substantial changes to the QP engine

Wavelets

- Mathematical tool for hierarchical decomposition of functions/signals
- Popular in the speech/signal/image processing domains

One-Dimensional Haar Wavelets

- Simplest wavelet basis, easy to understand and implement

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Averages</th>
<th>Detail Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>[2, 2, 2, 3, 5, 4, 4]</td>
<td>([0, -1, -1, 0])</td>
</tr>
<tr>
<td>2</td>
<td>[2, 2, 4, 4]</td>
<td>([0.5, 0])</td>
</tr>
<tr>
<td>1</td>
<td>[1.5, 4]</td>
<td>([-1.25])</td>
</tr>
<tr>
<td>0</td>
<td>([2.75])</td>
<td>([-1.25])</td>
</tr>
</tbody>
</table>

Haar wavelet decomposition: \([2.75, -1.25, 0.5, 0, 0, -1, -1, 0]\)

Wavelet Coefficients

- Hierarchical decomposition structure (a.k.a. "error tree")

Coefficient "Supports" 2.75

Original data
Wavelet-based Synopsis

- **Key idea**: use a compact subset of wavelet coefficients [Matias et al 98]
- **Steps**
  - Compute wavelet transform of cumulative data distribution
  - **Coefficient thresholding**: only b=|C| coefficients can be kept
    - Take largest coefficients in absolute normalized value
    - Divide coefficients at resolution j by $\sqrt{2^j}$
    - Optimal in overall Mean Squared (L2) Error
  - Greedy heuristic methods
    - Retain coefficients leading to large error reduction

Estimating Selectivity

- **Range Queries**: sel(a <= X <= b) = C'[b] - C'[a-1]
  - C' is the (approximate) “reconstructed” cumulative distribution
  - Time: O(min{b, logN}), where b = size of wavelet synopsis \( |N| \) size of domain
- At most \( \log N + 1 \) coefficients are needed to reconstruct any C value

Multi-Dimensional Haar Wavelets

- **Basic pairwise averaging and differencing** ideas carry over to multiple data dimensions
- Two basic methodologies -- no clear “winner”
  - *Standard* Haar decomposition [Stollnitz et al 96]
  - *Non-standard* Haar decomposition [Chakrabarti et al 00]

Wavelets: Summary

- Suitable for multi-resolution data analysis
- Experimentally seem to perform better than histograms
- Multi-dimensional wavelets need to be studied more
- As with histograms, wavelets as AQP systems require substantial changes to the QP engine

Sampling

- **Basic Idea**: A small random sample S of the data R often well-represents the entire data
  - For a fast approx answer, apply the query to S & "scale" the result
  - Use a small random sample of rows of the original database table to answer queries approximately

Sampling: Basics

- E.g., S is a 20% sample of R
  - Original Query: Select count(*) from R where R.a = 0
    - Count = 10
  - Rewritten Query: Select 5 * count(*) from S where S.a = 0
    - Est. count = 5 * 2 = 10
Non-Uniform/Biased Sampling

- Different data sampled at different rates
  - E.g., outliers sampled at a higher rate for better accuracy
- Each tuple \( j \) is selected for the sample \( S \) with some probability \( P_j \)
  - If selected, it is added to \( S \) along with its scale factor \( = 1/P_j \)

<table>
<thead>
<tr>
<th>( R.a )</th>
<th>10</th>
<th>10</th>
<th>50</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S.f )</td>
<td>( \frac{1}{3} )</td>
<td>( \frac{1}{3} )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{1}{2} )</td>
</tr>
</tbody>
</table>

Original Query: Select \( \text{sum}(R.a) \) from \( R \) where \( R.b < 5 \)
Rewritten Query: Select \( \text{sum}(S.a * S.sf) \) from \( S \) where \( S.b < 5 \)

Sum(\( R.a \)) = 130
\[ \text{Sum}(S.a*S.sf) = 10\times3 + 50\times2 = 130 \]

Error Estimation

- Unbiased estimation for \( \text{count, sum, avg} \) queries
  - I.e., the expected value of the answer is the actual answer
- Error in the answer depends on
  - Sample size
  - Variance in the data
- Error can be estimated with high probability
  - 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 probability \( \geq 90\% \)

\[ \text{Expected Error} = \sqrt{\frac{n}{N}} \text{Var} \]

Even confidence intervals can be estimated using Central Limit Theorem
- E.g., \( 54,000 \pm 600 \) with probability \( 95\% \)

Online vs. Pre-computed Sampling

- **Online:**
  - Continuous refinement of answers (online aggregation)
  - Online sampling is expensive in practice, unless data is randomly clustered on disk
  - Online Query Processing tutorial [Haas, Hellerstein 01]

- **Pre-computed:**
  - Seeing entire data is very helpful (provably & in practice)
  - (But must construct synopses for a family of queries)
  - Often faster: pre-computed synopses can reside in memory
  - Middleware: Can use with any DBMS, no special index

Precomputed Sampling Based Architecture for AQP

Uniform Random Sampling

- Pre-computation using Bernoulli Sampling
  - Expected sample size = \( fN \), where \( f \) is the sampling fraction
  - Add each tuple to \( S \) if random \( 0 \to 1 < f \)
  - Low overheads, no random data access

- Pre-Computation using Reservoir Sampling [Vitter 85]
  - Maintains a sample \( S \) of fixed-size \( fN \)
  - Add each new item to \( S \) with appropriate probability
  - If overfull, 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 \( S \)
Advantages/Disadvantages of Uniform Random Sampling

Advantages
- Simplicity
- Efficient preprocessing

Disadvantages
- Large errors due to
  - Large data variance
  - Low selectivity queries

Large Data Variance

Biased Sampling Techniques: Outlier Indexing

- Basic Idea [Chaudhuri et al 01]
  - Separate the outliers from the rest of the data into an outlier index
  - Keep a uniform random sample of the data
  - Use outlier index as well as random sample to answer queries

Outlier Indexing Scheme

Selection of Outlier Index.

- Select outliers (subject to space restriction) such that the rest of the relation has least variance.

Theorem: Given a sorted set of values the set that reduces the variation looks like:

..., v_{k-1}, v_k, v_{k+1}, v_{k+2}, ..., v_m, v_{m+1}, ...
### Algorithm Outlier Index

\[ 4, 6, 1, 10, 7 \]

\[ \text{sort} \]

\[ 1, 4, 6, 7, 10 \]

\[ \text{Outliers} = \{1, 10\} \]

\[ \text{Var} = 1.55 \]

\[ 1, 4, 6, 7, 10 \]

\[ \text{Var} = 2.88 \]

\[ \text{Var} = 1.55 \]

\[ \text{Var} = 4.22 \]

### Algorithm Implementation

- Requires a single pass over the relation.
- Main memory required = 2 * size of outlier index.

### Discussion

- How to allocate storage between samples and outlier index?
- Outlier index useful for *Sum, Avg*
  - Not useful for *Count, Max, Min*
- **Global Outliers versus Local Outliers**

### Biased Sampling Techniques: ICICLES [Ganti et al 00]

- Let \( W = \{q_1, q_2, \ldots\} \) be a query workload
- Biased sampling scheme exploits data locality
  - More focus (i.e., #sample points) in frequently-queried regions
- Biased sample incrementally maintained and adapts (“self-tune”) to changing workload

### Drawback with Static Uniform Random Samples

- In practice, queries exhibit locality
- Consequence: sample wastes precious real estate
- Example
  - Consider the relation recording sales in all Walmart outlets across United States
  - An analyst based in Madison may only be interested in sales in and around Wisconsin
  - In such cases, uniformly sampling the relation wastes space

### Locality in Query Workload

Workload \( W: Q_1, \ldots, Q_n \)

- \( W \) exhibits locality if, say, Wisconsin and Illinois tuples are accessed more often

A query \( Q \) follows \( W \) if tuples required to answer \( Q \) are accessed more frequently by queries in \( W \)
Problem
- Given
  - Relation R
  - Workload W: Q₁,…,Qₙ
  - A fixed amount of space
- Goal: Maintain a "sample Sₘ(W)(R) of R tuned w.r.t. W" to improve accuracy for queries following W

ICICLES
- R(Q): set of tuples in R required to answer Q
- Icicle Sₘ(W)(R): Random sample of R U⁺ R(Q₁) U⁺… U⁺ R(Qₙ)
  - Tuples required often are more likely to be in Sₘ(W)(R)

Maintaining Icicles
- Current workload W: Q₁,…,Qₙ
- L: R U⁺ R(Q₁) U⁺… U⁺ R(Qₙ)
- Current icicle Sₘ(W)(R): random sample of L
- New query: Q
- Update Sₘ(W)(R) to be the sample of L U⁺ R(Q)
  - Reservoir sampling technique to update
- Note: L is not materialized

Restricting the Workload
- Maintaining icicles requires queries to be answered exactly!
- Which queries to include in the workload?
  - Tradeoff between precision versus overhead
- Restrict the workload to such queries
  - Avoids extra overhead
  - Represents the most relevant set of queries

ICICLES: Discussion
- Quality Guarantee: The number of tuples in Sₘ(W)(R) satisfying a query Q is higher (than a static sample) if Q follows W
  - Theorem: If a query Avg(Q) follows the workload W, then the icicle Sₘ(W)(R) is expected to yield a better estimate for Avg(Q) than a static sample
- Queries with different counts have different errors
- Does not claim optimality
- Does not handle data variance

Biased Sampling Techniques: Congressional Sampling
- Approximating Group By queries: [Acharya et al 00]
  - Decision support queries routinely segment data into groups & then aggregate the information within each group
    - Each table has a set of “grouping columns”; queries can group by any subset of these columns
  - Goal: Maximize the accuracy for all groups (large or small) in each group-by query
Congressional Sampling
- E.g., census DB with state (s), gender (g), and income (i)
  - Q: Avg(i) group-by s: seek good accuracy for all 50 states
  - Q: Avg(i) group-by s, g: seek good accuracy for all 100 groups
- Congress Algorithm: Considers a "workload" of all subsets of grouping columns, and attempts to design a sample such that all groups get equal importance

Congress: Discussion
- Basically uses a form of stratified sampling
- Allocation of samples not necessarily optimal
- Does not consider data variance

Biased Sampling Techniques: STRAT
- Generalization of Outlier Indexing, ICICLES, and Congressional Sampling
  - [Chaudhuri, Das, Narasayya 01]
- Based on stratified sampling

Balancing Errors Across Multiple Queries
- Pay attention to:
  - "Hot spots" e.g., R_2
  - Small regions e.g., R_5

Intuition
- Stratification reduces data variance
- Allows "hot spots" in relation to be given more importance
  - Workload provides necessary information
  - Don't take workload literally

STRAT: an algorithm based on stratified sampling
- Different strata sampled at different rates
Lifting Workload to a Query Distribution

- Motivation - Resilience to variations in workload
- Let \( W = \{Q_1, ..., Q_q\} \) be the given workload
- We assume that \( W \) is a sample from a query distribution \( p_W \)
- Lifting \( \Rightarrow \) deriving \( p_W \) from \( W \)

Modeling Query Distribution

- Probability of \( Q' > Q'' \)
- \( \text{prob}(Q|W) = \text{parametric function of } Q \)'s overlap with queries in \( W \)
- Similar to kernel density estimation (queries in \( W \) = "sample points")

Steps in STRAT

- Stratification
  - \( R \) is partitioned into strata: \( \{R_1, ..., R_r\} \)
- Allocation
  - Total of \( k \) samples
  - Stratum \( R_j \) contributes \( k_j \) samples such that error for lifted workload is minimized
  - \( k_1 + ... + k_r = k \)

Stratification Step

- \( R_1, ..., R_5 \) are called fundamental regions

Allocation Step

- Assume \( r \) unknowns: \( k_1, ..., k_r \)
- Express error for lifted workload in terms of unknowns
- For e.g., for Mean Squared Error, \( \text{MSE}(p_{W'}) \):
  \[
  \text{MSE}(p_{W'}) = \sum_{s \subseteq \alpha_k} n_r^2 \delta(1-\delta) + \sum_{t \subseteq \beta_k} n_r^2 \gamma(1-\gamma)
  \]
  \[
  \left( \sum_{s \subseteq \alpha_k} \delta_n s + \sum_{t \subseteq \beta_k} \gamma_n t \right)^2
  \]
- Minimize MSE\( (p_W) \) subject to \( k_1 + k_2 + ... + k_r = k \)
Minimizing MSE($p_w$)

- Minimize MSE($p_w$) subject to $k_1 + k_2 + ... + k_r = k$
- Using calculus

$$k_j = k \left( \frac{\beta_j}{\sum \beta_j} \right)$$

STRAT: Summary

- Formal framework
  - Quantify errors for lifted workload
  - Frame it as an optimization problem
- Principled, pragmatic solution
  - Provide error estimates
- Unifies key ideas from past work
  - Weighted sampling, outlier indexing, congressional sampling
- Lifting workload to query distributions
  - Motivation: resilience to variations in workload

Dynamic Sample Selection

- Basic Idea [Babcock, Chaudhuri, Das 03]
  - Optimal bias differs from query to query
  - Past work: carefully select biased sample to give good answers for many queries
  - Instead, pre-compute many samples, and only use appropriate samples at run time

Improved accuracy, no change to query time
- Query time is the scarce resource
- OK to use extra pre-processing, disk space

Small vs. Large Groups

- In a group-by query, small groups are difficult to approximate
- Small Group Sampling
  - A specific example of dynamic sample selection for handling group by queries

How to pick a good set of samples?
- Construct many differently-biased samples
- For each query, use the best sample and ignore the others

Given a query, what's the best sample?
Small Group Sampling

Main idea: Treat small and large groups differently

- Large Groups: Use uniform random sample
  - Well-represented in sample
  - Good quality of approximation

Small Groups: Use Original Data

- Contain few records, by definition
- Thus can be scanned very quickly

Finding the Small Groups

- Heuristic idea:
  - Most small groups in most queries have a rare value for at least one grouping attribute
  - Small group in this query → rare value in entire DB
  - Not always true (snowblower sales in California)

Summary of Small Group Sampling:

- Identify rare values during pre-processing
- Store rows with rare values in a different (small) table
- At query time, scan small groups table for each grouping attribute

Pre-Processing Steps

- Create a table sample_all containing a uniform random sample of all data
- For each attribute A in the schema:
  - Identify rare values for attribute A
  - Create a table smGrps_A containing all records with rare A values
  - Size of smGrps_A table limited by threshold

Pre-Processing Steps

- Augment rows in sample_all, smGrps_* with table membership information
  - Some rows may be added to multiple tables
  - One extra bitmask column: which small group tables contain this row?
  - Used to avoid double-counting during query processing
Answering Queries Using Small Group Sampling

Values of attribute A

Common

Rare

Values of attribute B

Common

Rare

sample_all

smGrps_A

smGrps_B

Dynamic Sample Selection: Summary

- Dynamic Sample Selection
  - Gain accuracy at the cost of disk space.
  - Non-uniform samples are good, but different ones are good for different queries.
  - Build lots of different non-uniform samples.
  - For each query, pick the best sample.
- Small Group Sampling
  - Treat large and small groups differently.
  - Uniform sampling works well for large groups.
  - Small groups are cheap to scan in their entirety.

Query Answering Example

- Run query on small group table for each grouping attribute
- Run scaled query on \textit{sample\_all}
- Combine answers

```
SELECT A,B,COUNT(*) as cnt
FROM smGrps_A
WHERE C=10
GROUP BY A,B
UNION ALL
SELECT A,B,COUNT(*) as cnt
FROM smGrps_B
WHERE C=10 AND bitmask & 1 = 0
GROUP BY A,B
```

Sampling over multi-table databases

- Sampling over the joins of tables is difficult [Chaudhuri et al 99]
  - Not possible to push the sampling operator below the join
- However, sampling over foreign-key joins is ok
  - In star schemas, F-K joins represent a “widening” of the fact table

Sampling over F-K Joins

- Two approaches
  - Only pre-sample the fact table, and at run time join with the dimension tables
  - Join synopses, where we pre-sample from materialized F-K joins

Join Synopses for F-K Joins

- Based on sampling from materialized foreign key joins [Acharya et al 99]
  - Typically < 10% added space required
  - Yet, can be used to get a uniform sample of ANY foreign key join
  - Plus, fast to incrementally maintain
Distinct Value Estimation

- Example of distinct value query
  
  ```sql
  select count(distinct target-attr) 
  from T 
  where P
  ```

- Uniform Sampling-based approaches
  - Collect and store uniform sample. At query time, apply predicate to sample. Estimate based on a function of the distribution. Extensive literature.

Distinct Value Estimation

- Pessimistic lower bound [Chaudhuri et al 00]
  
  For any $p > e^{-r}$ there exists a relation such that with probability at least $p$,
  
  $$\text{error}(d^*) > \left[ n \ln \left( \frac{n}{r} \right) \right]$$

  I.e., any estimator must examine (sample) almost the entire table to guarantee the estimate is within a factor of 10 with probability $\geq 1/2!$

“Proof” of Lower Bound

- Consider two distributions, one with all identical values, and another with $x+1$ distinct values, of which $x$ are singleton and the $(x+1)^{st}$ occurs $n-x$ times

- A small sample will miss the singleton values with high probability, and hence cannot distinguish between the two distributions

Example of a DV Estimator

- Let $r$ be the sample size from $n$ tuples
- Let $f_i$, for $1 \leq i \leq r$, denote the number of distinct values that occur exactly $i$ times. (Clearly $\sum f_i = r$)

- Estimator is
  
  $$d' = \sqrt{\frac{n}{r}} f_1 + \sum_{2 \leq i \leq r} f_i$$

Sampling-Based Distinct Value Estimation: Summary

- Very difficult to solve using sampling
  - Often, some data distribution information needs to be used for good results

- One pass hash-based approaches more successful
  - AQP over data streams
  - Not topic of this tutorial

Sampling for AQP: Summary

- Resilient to high dimensional data
- Easy to give probabilistic error guarantees
- Can be implemented with minimal changes to QP
**AQP topics not covered in this tutorial**

- Other synopses models for joint distributions
  - Kernel Density models, Bayes Nets
- AQP over data streams
- Beyond aggregations
  - How to return approximations to set-valued results

**AQP Systems in the Industry**

- AQP internally used by query optimizers for selectivity estimation
- AQP not yet externalized by major vendors
  - Although sampling operators are appearing in commercial DBMS
- Research prototypes, e.g.
  - AQP from MSR
  - AQUA from Bell Labs

**Conclusions**

- Much theoretical advance in AQP algorithms
- Big gap between algorithms and actual systems
  - Future work should focus on this aspect
- But, likely to catch on as data repositories are ever increasing