Outline

- Intro & Approximate Query Answering Overview
  - Synopses, System architecture, 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
- Advanced Techniques & Future Directions
  - Streaming Data, Dependency-based, Work-load tuned
- Conclusions
**Introduction & Motivation**

- 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 are the total sales of product X in NJ?"
  - 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
  
  $\text{59,000 +/- 500 (with 95\% confidence) in 10 seconds}$
  
  vs.
  
  $\text{59,152.25 in 10 minutes}$

- Achieved by answering the query based on samples or other synopses of the data
- Speed-up obtained because synopses are 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

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

The Aqua Architecture [GMP97a, AGP99]

- Picture with Aqua: Aqua is middleware that sits 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 approx 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
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
- Advanced Techniques & Future Directions
- Conclusions

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])
- **Can maintain** using one-pass algorithms (insertions only), or
- Use a backing sample [GMP97b]: Keep bucket counts up-to-date
  - Merge adjacent buckets with small counts
  - Split any bucket with a large count, using the sample to select a split value (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: 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

1-D Histograms

- Answering queries from histograms:
  - (Implicitly) map the histogram back to an approximate relation, apply the query to the approximate relation
  - Continuous value mapping [SAC79]:
    - Count spread evenly among bucket values
  - Uniform spread mapping [PIH96]:
    - Need number of distinct in each bucket
**1-D Histograms: V-Optimal**

- [IP95] defined V-optimal & showed it minimizes the average selectivity estimation error for equality-joins & selections
  - Select buckets to minimize frequency variance within buckets
- [JKM98] gave an $O(B^2N^2)$ time dynamic programming algorithm
  - $F[k] = \text{freq. of value } k$; $\text{AVGF}[i:j] = \text{avg freq for values } i..j$
  - $\text{SSE}[i:j] = \sum_{k=i}^{j} (F[k]^2 - (j-i+1) \times \text{AVGF}[i:j]^2)$
  - 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 $\text{SSE}[i:j]$ in constant time
  - Let $SSEP(i,k) = \min \text{SSE for } F[1..i] \text{ using } k \text{ buckets}$
  - Then $SSEP(i,k) = \min_{j=1..i-1} (SSEP(j,k-1) + \text{SSE}[j+1:i])$, i.e., suffices to consider all possible left boundaries for $k$th bucket
  - Also gave faster approximation algorithms

**Self-Tuning 1-D Histograms**

- Tune Bucket Frequencies:
  - Compare actual selectivity to histogram estimate
  - Use to adjust bucket frequencies
  - Divide $d \times \text{Error}$ proportionately, $d =$ dampening factor
  - Actual = 60
  - Estimate = 40
  - Error = +20
  - $d = \frac{1}{2}$ of Error = +10
  - So divide +4,+3,+3
**Self-Tuning 1-D Histograms**

2. Restructure:
   - Merge buckets of near-equal frequencies
   - Split large frequency buckets

![Histogram Diagram]

Extends to Multi-D

---

**Sampling: Basics**

- Idea: A small random sample \( S \) of the data often well-represents the entire data
  - For a fast approx answer, apply the query to \( S \) & “scale” the result
  - E.g., \( S \) is a 20% sample
    ```sql
    select count(*) from R where R.a = 0
    select 5 * count(*) from S where S.a = 0
    ```
  - 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 ± 600 with probability ≥ 90%
### Sampling: Confidence Intervals

<table>
<thead>
<tr>
<th>Method</th>
<th>90% Confidence Interval ($\epsilon$)</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)

$\sigma(R)$ = standard deviation of the values of R.A; $\sigma(S) = s.d.$ 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 $P_j$ (can depend on values in tuple $j$)
  - If selected, it is added to $S$ along with its scale factor $sf = 1/P_j$

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

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

<table>
<thead>
<tr>
<th>$R.a$</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>50</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_j$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>$S.sf$</td>
<td>---</td>
<td>3</td>
<td>---</td>
<td>---</td>
<td>2</td>
</tr>
</tbody>
</table>

| $\sum(R.a)$ | 130 |
| $\sum(S.a * S.sf)$ | 10*3 + 50*2 = 130 |
One-Dimensional Haar Wavelets

- Wavelets: mathematical tool for hierarchical decomposition of functions/signals
- Haar wavelets: simplest wavelet basis, easy to understand and implement
  - Recursive pairwise averaging and differencing at different resolutions

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

Haar Wavelet Coefficients

- Hierarchical decomposition structure (a.k.a. “error tree”)

Coefficient “Supports”

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>“Supports”</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.75</td>
<td>+</td>
</tr>
<tr>
<td>-1.25</td>
<td>+</td>
</tr>
<tr>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>-1</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
Wavelet-based Histograms [MVW98]

- Problem: range-query selectivity estimation
- Key idea: use a compact subset of Haar/linear wavelet coefficients for approximating the data distribution
- Steps
  - compute cumulative data distribution \( C \)
  - compute Haar (or linear) wavelet transform of \( C \)
  - coefficient thresholding: only \( b \ll |C| \) coefficients can be kept
    - take largest coefficients in absolute normalized value
      - Haar basis: divide coefficients at resolution \( j \) by \( \sqrt{2^j} \)
      - Optimal in terms of the overall Mean Squared (L2) Error
    - Greedy heuristic methods
      - Retain coefficients leading to large error reduction
      - Throw away coefficients that give small increase in error

Using Wavelet-based Histograms

- Selectivity estimation: \( \text{sel}(a \leq X \leq b) = C[b] - C[a-1] \)
  - \( C \) is the (approximate) “reconstructed” cumulative distribution
  - Time: \( O(\min(b, \log N)) \), where \( b \) = size of wavelet synopsis (no. of coefficients), \( N \) = size of domain
  - At most \( \log N + 1 \) coefficients are needed to reconstruct any \( C \) value

- Empirical results over synthetic data
  - Improvements over random sampling and histograms (MaxDiff)
**Dynamic Maintenance of Wavelet-based Histograms [MVW00]**

- Build Haar-wavelet synopses on the original data distribution
  - Similar accuracy with CDF, makes maintenance simpler
- Key issues with dynamic wavelet maintenance
  - Change in single distribution value can affect the values of many coefficients (path to the root of the decomposition tree)
    - As distribution changes, “most significant” (e.g., largest) coefficients can also change!
      - Important coefficients can become unimportant, and vice-versa

**Effect of Distribution Updates**

- Key observation: for each coefficient $c$ in the Haar decomposition tree
  - $c = (\text{AVG(leftChildSubtree}(c)) - \text{AVG(rightChildSubtree}(c)) ) / 2$

Only coefficients on path($d$) are affected and each can be updated in constant time.
**Maintenance Architecture**

- “Shake up” when log reaches max size: for each insertion at d
  - for each coefficient c on path(d) and in H': update c
  - for each coefficient c on path(d) and not in H or H':
    - insert c into H' with probability proportional to $1/2^h$, where h is the “height” of c (Probabilistic Counting [FM85])
    - Adjust H and H' (move largest coefficients to H)

**Outline**

- Intro & Approximate Query Answering Overview
- One-Dimensional Synopses
- Multi-Dimensional Synopses and Joins
  - Multi-dimensional Histograms
  - Join sampling
  - Multi-dimensional Haar Wavelets
- Set-Valued Queries
- Advanced Techniques & Future Directions
- Conclusions
Multi-dimensional Data Synopses

- Problem: Approximate the joint data distribution of multiple attributes
  - Motivation
    - Selectivity estimation for queries with multiple predicates
    - Approximating OLAP data cubes and general relations
  - Conventional approach: Attribute-Value Independence (AVI) assumption
    - \( \text{sel}(p(A1) \& p(A2) \& \ldots) = \text{sel}(p(A1)) \times \text{sel}(p(A2)) \times \ldots \)
    - Simple -- one-dimensional marginals suffice
    - BUT: almost always inaccurate, gross errors in practice (e.g., [Chr84, FK97, Poo97])

Multi-dimensional Histograms

- Use small number of multi-dimensional buckets to directly approximate the joint data distribution
- Uniform spread & frequency approximation within buckets
  - \( n(i) = \text{no. of distinct values along } A_i, F = \text{total bucket frequency} \)
  - approximate data points on a \( n(1) \times n(2) \times \ldots \) uniform grid, each with frequency \( F / (n(1) \times n(2) \times \ldots) \)
**Multi-dimensional Histogram Construction**

- Construction problem is much harder even for two dimensions [MPS99]
- **Multi-dimensional equi-depth histograms** [MD88]
  - Fix an ordering of the dimensions $A_1, A_2, \ldots, A_k$, let $\alpha = k$th root of desired no. of buckets, initialize $B = \{\text{data distribution}\}$
  - For $i=1, \ldots, k$: Split each bucket in $B$ in $\alpha$ equi-depth partitions along $A_i$; return resulting buckets to $B$
  - Problems: limited set of bucketizations; fixed $\alpha$ and fixed dimension ordering can result in poor partitionings

- **MHIST-p histograms** [PI97]
  - 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

---

**Equi-depth vs. MHIST Histograms**

**Equi-depth (a1=2,a2=3) [MD88]**

```
<table>
<thead>
<tr>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>280</td>
</tr>
</tbody>
</table>
```

**MHIST-2 (MaxDiff) [PI97]**

```
<table>
<thead>
<tr>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>280</td>
</tr>
</tbody>
</table>
```

- MHIST: choose bucket/dimension to split based on its criticality; allows for much larger class of bucketizations (hierarchical space partitioning)
- Experimental results verify superiority over AVI and equi-depth
Other Multi-dimensional Histogram Techniques -- GENHIST [GKT00]

- Key idea: allow for overlapping histogram buckets
  - Allows for a much larger no. of distinct frequency regions for a given space budget (= #buckets)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

\[ a+b+c+d \]

- 9 distinct frequencies (13 if different-size buckets are used)

- Greedy construction algorithm: Consider increasingly-coarser grids
  - At each step select the cell(s) c of highest density and move enough randomly-selected points from c into a bucket to make c and its neighbors "close-to-uniform"
  - Truly multi-dimensional "split decisions" based on tuple density
    -- unlike MHIST

Other Multi-dimensional Histogram Techniques -- STHoles [BCG01]

- Multi-dimensional, workload-based histograms
  - Allow bucket nesting (rather than arbitrary overlap) -- "bucket tree"
  - Intercept query result stream and count \(|q \cap b|\) for each bucket b (≤ 10% overhead in MS SQL Server 2000)
  - Drill "holes" in b for regions of different tuple density and "pull" them out as children of b (first-class buckets)
  - Consolidate/merge buckets of similar densities (keep #buckets constant)
**Sampling for Multi-D Synopses**

- Taking a sample of the rows of a table captures the correlations in those (and only those) rows
  - Answers are unbiased & confidence intervals apply
  - Thus guaranteed accuracy for count, sum, and average queries on single tables, as long as the query not too selective

- Problem with joins [AGP99,CMN99]:
  - Join of two uniform samples is not a uniform sample of the join
  - Join of two samples typically has very few tuples

![Diagram of Foreign Key Join](image)

**Join Synopses for F-Key Joins**

- Based on sampling from materialized foreign key joins
  - Typically < 10% added space required
  - Yet, can be used to get a uniform sample of ANY foreign key join
  - Plus, fast to incrementally maintain

- Significant improvement over using just table samples
  - E.g., for TPC-H query Q5 (4 way join)
    - 1%-6% relative error vs. 25%-75% relative error, for synopsis size = 1.5%, selectivity ranging from 2% to 10%
    - 10% vs. 100% (no answer!) error, for size = 0.5%, select. = 3%

[AGP99]
Multi-dimensional Haar Wavelets

- Basic “pairwise averaging and differencing” ideas carry over to multiple data dimensions
- Two basic methodologies -- no clear “winner” [SDS96]
  - Standard Haar decomposition
  - Non-standard Haar decomposition

Discussion here: focus on non-standard decomposition
- See [SDS96, VW99] for more details on standard Haar decomposition
- [MVW00] also discusses dynamic maintenance of standard multi-dimensional Haar wavelet synopses

Two-dimensional Haar Wavelets -- Non-standard decomposition

\[
\begin{array}{cccc}
  c_3 & d_3 & c_4 & d_4 \\
  a_3 & b_3 & a_4 & b_4 \\
  c_1 & d_1 & c_2 & d_2 \\
  a_1 & b_1 & a_2 & b_2 \\
\end{array}
\]

\[
A_1 = \frac{a_1+b_1+c_1+d_1}{4}
\]

Detail coeff = \frac{(a_1+b_1-c_1-d_1)}{4}

Detail coeff = \frac{(a_1-c_1+b_1-d_1)}{4}

Detail coeff = \frac{(a_1-b_1-c_1+d_1)}{4}

A = \frac{(A_1+A_2+A_3+A_4)}{4}

Detail coeff = \frac{(A_1+A_2-A_3-A_4)}{4}

Detail coeff = \frac{(A_1-A_2+A_3-A_4)}{4}

Detail coeff = \frac{(A_1-A_2-A_3+A_4)}{4}
Two-dimensional Haar Wavelets --
Non-standard decomposition

Wavelet Transform Array:

After averaging and differencing

After distributing results

Final wavelet transform array
Non-standard Two-dimensional Haar Basis -- Coefficient Supports

Constructing the Wavelet Decomposition

<table>
<thead>
<tr>
<th>Joint Data Distribution Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr1 0</td>
</tr>
<tr>
<td>Attr2 0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Relation (ROLAP) Representation

<table>
<thead>
<tr>
<th>Attr1</th>
<th>Attr2</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

- Joint data distribution can be very sparse!
- Key to I/O-efficient decomposition algorithms: Work off the ROLAP representation
  - Standard decomposition [VW99]
  - Non-standard decomposition [CGR00]
- Typically require a small (logarithmic) number of passes over the data
Range-sum Estimation Using Wavelet Synopses

- Coefficient thresholding
  - As in 1-d case, normalizing by appropriate constants and retaining the largest coefficients minimizes the overall L2 error
- Range-sums: selectivity estimation or OLAP-cube aggregates [VW99] ("measure attribute" as count)
- Only coefficients with support regions intersecting the query hyper-rectangle can contribute
  - Many contributions can cancel each other [CGR00, VW99]

Outline

- Intro & Approximate Query Answering Overview
- One-Dimensional Synopses
- Multi-Dimensional Synopses and Joins
- Set-Valued Queries
  - Using Histograms
  - Using Samples
  - Using Wavelets
- Advanced Techniques & Future Directions
- Conclusions
Approximating Set-Valued Queries

- Problem: Use synopses to produce "good" approximate answers to generic SQL queries -- selections, projections, joins, etc.
  - Remember: synopses try to capture the joint data distribution
  - Answer (in general) = multiset of tuples
- Unlike aggregate values, NO universally-accepted measures of "goodness" (quality of approximation) exist

<table>
<thead>
<tr>
<th>Query Answer</th>
<th>Subset Approximation (e.g., from 20% sample)</th>
<th>“Better” Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error Metrics for Set-Valued Query Answers

- Need an error metric for (multi)sets that accounts for both
  - differences in element frequencies
  - differences in element values
- Traditional set-comparison metrics (e.g., symmetric set difference, Hausdorff distance) fail

- Proposed Solutions
  - MAC (Match-And-Compare) Error [IP99]: based on perfect bipartite graph matching
  - EMD (Earth Mover’s Distance) Error [CGR00, RTG98]: based on bipartite network flows
Using Histograms for Approximate Set-Valued Queries [IP99]

- Store histograms as relations in a SQL database and define a histogram algebra using simple SQL queries
- Implementation of the algebra operators (select, join, etc.) is fairly straightforward
  - Each multidimensional histogram bucket directly corresponds to a set of approximate data tuples
- Experimental results demonstrate histograms to give much lower MAC errors than random sampling

- Potential problems
  - For high-dimensional data, histogram effectiveness is unclear and construction costs are high [GKT00]
  - Join algorithm requires expanding into approximate relations
    - Can be as large (or larger!) than the original data set

Approximate Query Processing Using Wavelets [CGR00]

- Reduce relations into compact wavelet-coefficient synopses

**Entire query processing in the compressed (wavelet) domain**
Wavelet Query Processing

- Each operator (e.g., select, project, join, aggregates, etc.)
  - input: set of wavelet coefficients
  - output: set of wavelet coefficients

- Finally, rendering step
  - input: set of wavelet coefficients
  - output: (multi)set of tuples

Selection -- Relational Domain

- In relational domain, interested in only those cells inside query range
- In wavelet domain, interested in only the coefficients that contribute to those cells
Selection -- Wavelet Domain

Equi-join -- Relational Domain

- **Relational domain:** Join count $= 7 \times 3 = (A1-A3)(B2+B3)$
- **Wavelet domain:** $A1 \times B2 + A1 \times B3 - A3 \times B2 - A3 \times B3$
- **Consider all pairs of coefficients:** (1) check joinability (overlap in join dimension(s)), (2) compute output coefficients
Equi-join -- Wavelet Domain

Join output coefficient:

Set-Valued Queries via Samples

- Applying the set-valued query to the sampled rows, we very often obtain a **subset of the rows in the full answer**
  - E.g., Select all employees with 25+ years of service
  - Exceptions include certain queries with nested subqueries (e.g., select all employees with above average salaries: but the average salary is known only approximately)

- Extrapolating from the sample:
  - Can treat each sample point as the center of a cluster of points
  - Alternatively, Aqua [GMP97a, AGP99] returns an approximate count of the number of rows in the answer and a representative subset of the rows (i.e., the sampled points)
    - Keeps result size manageable and fast to display
Outline

- Intro & Approximate Query Answering Overview
- One-Dimensional Synopses
- Multi-Dimensional Synopses and Joins
- Set-Valued Queries
- Advanced Techniques & Future Directions
  - Biased/Stratified/Congressional Sampling
  - Distinct-value queries
  - Dependency-based synopses
  - Streaming Data
- Conclusions

Biased Sampling Techniques -- ICICLES [GLR00]

- Biased sampling scheme that dynamically adapts to query workload
  - Exploit data locality -- more focus (i.e., #sample points) in frequently-queried regions
- Let Q = {q1, q2, . . .} be a query workload, R(qi) = subset of R used in answering query qi
  - L(R, Q) = Extension of R wrt Q = \( R \bigcup_{qi \in Q} R(qi) \) (multiset of tuples)

- Icicle: Uniform random sample of L(R,Q)
- Incrementally maintained and adapt (“self-tune”) to workload through Reservoir Sampling technique [Vit85]
- Unbiased Icicle estimators: New formulas to account for duplicates and bias in sample selection
- Provably better (smaller variance) than uniform for “focused” queries (that follow the workload model)
Biased Sampling Techniques --
Stratified Samples [CDN01]

• Formulate sample selection as an optimization problem
  - Minimize query-answering error for a given workload model
• Technique for "lifting a fixed workload W" to produce a probability distribution over all possible queries
  - Similar to kernel density estimation (queries in W = "sample points")

\[ W = \{ q_1, q_2 \} \]

"Fundamental regions" induced by W

\[ q_1 \]
\[ q_2 \]
\[ q \]

\[ \text{prob}(q|W) = \text{parametric function of } q \text{'s overlap with queries in } W \]

• Problem: Find sample of size $k$ that minimizes expected error for a given "lifted" workload
• Solution: Stratified sampling [Coc77]
  - Collection of uniform samples (of total size $k$) over disjoint subsets ("strata") of the population
  - Much better estimates when variance within strata is small [Coc77]
• Stratification: Selecting appropriate partitioning of $R$
  - Using "fundamental regions" as strata is optimal for COUNT
  - For SUM, partition "fundamental regions" further to reduce variance of the aggregated attribute (Neymann technique [Coc77])
• Allocation: Breaking $k$ among strata
  - Closed form solutions (valid under certain simplifying assumptions)
Synopses for Group-By Queries

- 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
  - 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

- Technique: Congressional Samples [AGP00]
  - House: Uniform sample: good for when no group-by
  - Senate: Same size sample per group when use all grouping columns: good for queries with all columns
  - Congress: Combines House & Senate, but considers all subsets of grouping columns, and then scales down

Distinct Values Queries

- select count(distinct target-attr)
  - from rel
  - where P

- select count(distinct o_custkey)
  - from orders
  - where o_orderdate >= '2001-01-01'

  - How many distinct customers have placed orders this year?

- Includes: column cardinalities, number of species, number of distinct values in a data set / data stream

Template

TPCH example
Distinct Values Queries

• 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 (see, e.g., [CCM00])
    • Many functions proposed, but estimates are often inaccurate
    • [CCM00] proved must examine (sample) almost the entire table to guarantee the estimate is within a factor of 10 with probability > 1/2, regardless of the function used!

• One pass approaches
  - A hash function maps values to bit position according to an exponential distribution [FM85] (cf. [Coh97,AMS96])
    • 00001011111 estimate based on rightmost 0-bit
    • Produces a single count: Does not handle subsequent predicates

Distinct Values Queries

• One pass, sampling approach: Distinct Sampling [Gib01]:
  - A hash function assigns random priorities to domain values
  - Maintains $O(\log(1/\delta)/\varepsilon^2)$ highest priority values observed thus far, and a random sample of the data items for each such value
  - Guaranteed within $\varepsilon$ relative error with probability $1 - \delta$

  - Handles ad-hoc predicates: E.g., How many distinct customers today vs. yesterday?
    • To handle q% selectivity predicates, the number of values to be maintained increases inversely with q (see [Gib01] for details)

  - Good for data streams: Can even answer distinct values queries over physically distributed data. E.g., How many distinct IP addresses across an entire subnet? (Each synopsis collected independently!)

  - Experimental results: 0-10% error vs. 50-250% error for previous best approaches, using 0.2% to 10% synopses
Approximate Reports

- Distinct sampling also provides fast, highly-accurate approximate answers for report queries arising in high-volume, session-based event recording environments.

- **Environment**: Record events, produce precanned reports
  - Many overlapping sessions: multiple events comprise a session (single IP flow, single call set-up, single customer service call)
  - Events are time-stamped and tagged with session id, and then dumped to append-only databases
  - Logs sent to central data warehouse. Precanned reports executed every minute or hour. TPC-R benchmark

- Must maintain a uniform sample of the sessions & all the events in those sessions in order to produce good approximate reports. Distinct sampling provides this. Improves accuracy by factor of 10+

Dependency-based Histogram Synopses [DGR01]

- Extremes in terms of the underlying correlations!!
- **Dependency-Based (DB) Histograms**: explore space between extremes by explicitly identifying data correlations/independences
  - Build a *statistical interaction model* on data attributes
  - Based on the model, build a collection of low-dimensional histograms
  - Use this histogram collection to provide approximate answers
- **General methodology**, also applicable to other synopsis techniques (e.g., wavelets)
More on DB Histograms

- Identify (end exploit) attribute correlation and independence
  - Partial Independence:
    \[ p(\text{salary}, \text{height}, \text{weight}) = p(\text{salary}) \times p(\text{height}, \text{weight}) \]
  - Conditional Independence:
    \[ p(\text{salary}, \text{age} | YPE) = p(\text{salary} | YPE) \times p(\text{age} | YPE) \]
- Use forward selection to build a decomposable statistical model [BFH75], [Lau96] on the attributes
  - \( A, D \) are conditionally independent given \( B, C \)
    \[ p(\text{AD}|\text{BC}) = p(\text{A}|\text{BC}) \times p(\text{D}|\text{BC}) \]
  - Joint distribution
    \[ p(\text{ABCD}) = p(\text{ABC}) \times p(\text{BCD}) / p(\text{BC}) \]
- Build histograms on model cliques
- Significant accuracy improvements over pure MHIST
- More details, construction & usage algorithms, etc. in the paper 😊

Data Streams

- Data is continually arriving. Collect & maintain synopses on the data. Goal: Highly-accurate approximate answers
  - State-of-the-art: Good techniques for narrow classes of queries
  - E.g., Any one-pass algorithm for collecting & maintaining a synopsis can be used effectively for data streams
- Alternative scenario: A collection of data sets. Compute a compact sketch of each data set & then answer queries (approximately) comparing the data sets
  - E.g., detecting near-duplicates in a collection of web pages: Altavista
  - E.g., estimating join sizes among a collection of tables [AGM99]
Looking Forward...

- Optimizing queries for approximation
  - e.g., minimize length of confidence interval at the plan root
- Exploiting mining-based techniques (e.g., decision trees) for data reduction and approximate query processing
  - see, e.g., [BGR01], [GTK01], [JMN99]
- Dynamic maintenance of complex (e.g., dependency-based [DGR01] or mining-based [BGR01]) synopses
- Synopsis construction and approximate query processing over continuous data streams
  - see, e.g., [GKS01a], [GKS01b], [GKM01b]

Conclusions

- Commercial data warehouses: approaching several 100's TB and continuously growing
  - Demand for high-speed, interactive analysis (click-stream processing, IP traffic analysis) also increasing
- Approximate Query Processing
  - "Tame" these Terabytes and satisfy the need for interactive processing and exploration
  - Great promise
  - Commercial acceptance still lagging, but will most probably grow in coming years
  - Still looots of interesting research to be done!!
References (1)

  - Proposes exploiting simple (differential and combinational) data dependencies for effectively compressing data tables.
References (2)


References (3)

References (4)

  - Proposes the use of "multifractals" (i.e., 80/20 laws) to more accurately approximate the frequency distribution within histogram buckets.
  - Presents algorithms for building “range-optimal” histogram and wavelet synopses; that is, synopses that try to minimize the total error over all possible range queries in the data domain.
  - Proposes the “concise sample” and “counting sample” techniques for improving the accuracy of sampling-based estimation for a given amount of space for the sample synopsis.

References (5)

  - Proposes the “concise sample” and “counting sample” techniques for improving the accuracy of sampling-based estimation for a given amount of space for the sample synopsis.
References (6)

  - Proposes novel, Bayesian-network-based techniques for approximating joint data distributions in relational database systems.


  - Proposes and evaluates several sampling-based estimators for the number of distinct values in an attribute column.


References (7)


  - The above three papers propose and study serial histograms (i.e., histograms that bucket neighboring frequency values, and exploit results from majorization theory to establish their optimality wrt minimizing (extreme cases of) the error in multi-join queries.


  - Discusses the use of "fascicles" (i.e., approximate data clusters) for the semantic compression of relational data.

References (8)

- Proposes the use of SVD techniques for obtaining fast approximate answers from large time-series databases.
    - Proposes the use of linear splines to better approximate the data and frequency distribution within histogram buckets.
    - Proposes the use of the Discrete Cosine Transform (DCT) for compressing the information in multi-dimensional histogram buckets.
    - Proposes techniques for enhancing hierarchical multi-dimensional index structures to enable approximate answering of aggregate queries with progressively improving accuracy.
    - Presents an adaptive, sequential sampling scheme for estimating the selectivity of relational equi-join operators.
    - Presents adaptive-sampling-based techniques and estimators for approximating the result size of a relational projection operation.

References (9)

  - Presents adaptive-sampling-based techniques and estimators for approximating the result size of a relational projection operation.
References (10)

- Discusses the use of mixture models composed of multi-variate Gaussians for building compact models of OLAP data cubes and approximating range-sum query answers.

References (11)

Additional Resources

• Related Tutorials
    • http://www.research.att.com/~drknow/pubs.html
    • http://control.cs.berkeley.edu/sigmod01/
    • http://atlas.eml.org/ICDE/index_html

• Research Project Homepages
  - The AQUA and NEMESIS projects (Bell Labs)
    • http://www.bell-labs.com/project/{aqua, nemesis}/
  - The CONTROL project (UC Berkeley)
    • http://control.cs.berkeley.edu/
  - The Approximate Query Processing project (Microsoft Research)
    • http://www.research.microsoft.com/research/dmx/ApproximateQP/
  - The Dr. Know project (AT&T Research)
    • http://www.research.att.com/~drknow/