A Quick Introduction to Approximate Query Processing Part-III

CS286, Spring'2007

Minos Garofalakis

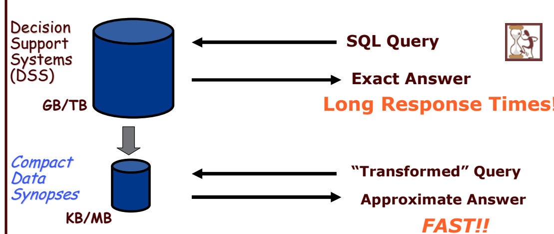
Decision Support Systems

- 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!)

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 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

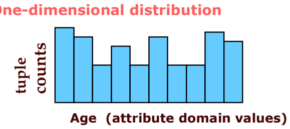
Approximate Query Processing using Data Synopses



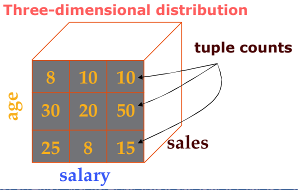
- How to construct effective *data synopses* ??

Relations as Frequency Distributions

One-dimensional distribution



Three-dimensional distribution



name	age	salary	sales
MG	34	100K	25K
JG	33	90K	30K
RR	40	190K	55K
JH	36	110K	45K
MF	39	150K	50K
DD	45	150K	50K
JN	43	140K	45K
AP	32	70K	20K
EM	24	50K	18K
DW	24	50K	28K

Outline

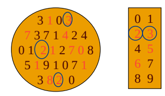
- Intro & Approximate Query Answering Overview
  - Synopses, System architectures, Commercial offerings
- One-Dimensional Synopses
  - **Histograms:** Equi-depth, Compressed, V-optimal, Incremental maintenance, Self-tuning
  - **Samples:** Basics, Sampling from DBs, Reservoir Sampling
  - **Wavelets:** 1-D Haar-wavelet histogram construction & maintenance
- Multi-Dimensional Synopses and Joins
- Set-Valued Queries
- Discussion & Comparisons
- Advanced Techniques & Future Directions

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
- Discussion & Comparisons
- Advanced Techniques & Future Directions
  - Dependency-based, Workload-tuned, Streaming data

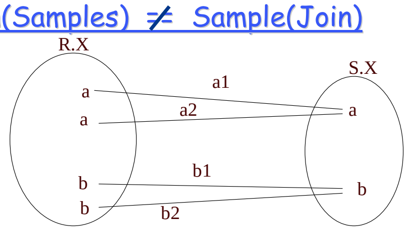
Sampling for Multi-D Synopses

- Taking a sample of the rows of a table captures the attribute correlations in 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 is 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



Foreign Key Join
40% Samples in Red
Size of Actual Join = 30
Size of Join of samples = 3

Join(Samples) ≠ Sample(Join)



- Join result = {a1, a2, b1, b2}
- Probability for a base tuple to be selected = 1/r
- Prob[select a1 and a2] = 1/r^3
- Prob[select a1 and b1] = 1/r^4

Small Results for Join(samples)

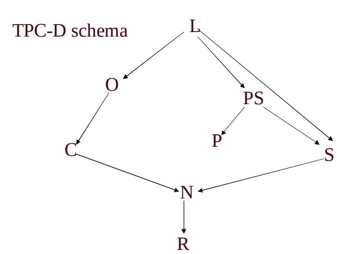
- Foreign key join of R and S (R→S)
  - Join result size = |R|
- 1% sample from both R and S ➔ 0.01% sample from the join result!!
  - Each tuple from sample(R) joins with a *single tuple from S*
  - Probability that tuple is kept is only 1% !

Join Synopses for Foreign-Key Joins [AGP99]

- 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%

Join Synopses

- *Schema-based sample summaries* from FK join results



Join Synopses: Key Observations

R1 → R2 → … → Rk

"Source relation"

- *One-to-one correspondence* between tuples in source relation and those in result of chain of FK-joins
- *Sample(R1) joined with R2, …, Rk = sample(FK-join chain)*
- To get a sample of a subchain of FK-joins "rooted" at source, just project away irrelevant attributes!
- Join synopses = set of such sample joins *for every source* and *maximal FK-join-chain* in the schema!
  - Can be used to answer **ANY FK-join query** over the given schema!

Join Synopses: Optimizations and Maintenance

R1 → R2 → … → Rk

"Source relation"

- Propose techniques for allocating space across join-synopses in order to minimize overall error metrics
- Incremental maintenance is easy, using "reservoir-sampling"-style techniques

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

c3	d3	c4	d4
a3	b3	a4	b4
c1	d1	c2	d2
a1	b1	a2	b2

A1 = (a1+b1+c1+d1)/4

Detail coeff = (a1+b1-c1-d1)/4

Detail coeff = (a1-b1+c1-d1)/4

Detail coeff = (a1-b1-c1+d1)/4

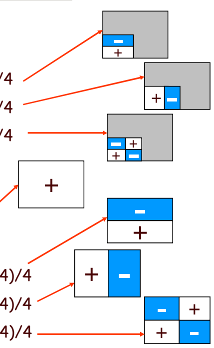
A3	A4
A1	A2

A = (A1+A2+A3+A4)/4

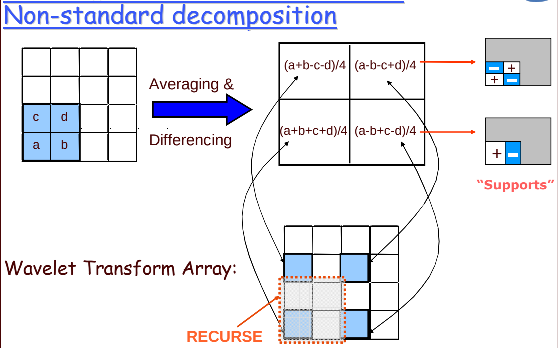
Detail coeff = (A1+A2-A3-A4)/4

Detail coeff = (A1-A2+A3-A4)/4

Detail coeff = (A1-A2-A3+A4)/4



Two-dimensional Haar Wavelets -- Non-standard decomposition



Two-dimensional Haar Wavelets -- Non-standard decomposition

Data Array

3	4	3	4
1	2	1	2
3	4	3	4
1	2	1	2

After averaging and differencing

-1	0	-1	0
2.5	-.5	2.5	-.5
-1	0	-1	0
2.5	-.5	2.5	-.5

After distributing results

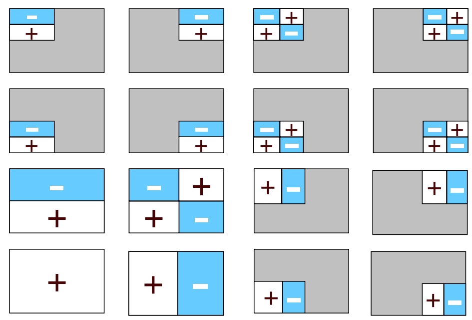
-1	-1	0	0
-1	-1	0	0
2.5	2.5	-.5	-.5
2.5	2.5	-.5	-.5

RECURSE

Final wavelet transform array

-1	-1	0	0
-1	-1	0	0
0	0	-.5	-.5
2.5	0	-.5	-.5

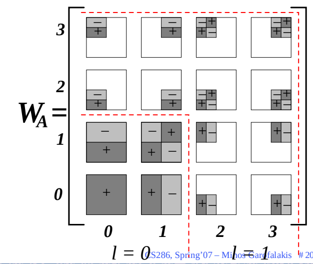
Non-standard Two-dimensional Haar Basis -- Coefficient Supports



Multi-dimensional Haar Wavelets

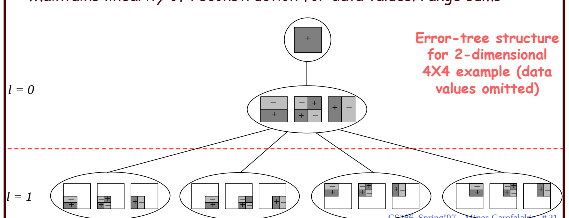
- Haar decomposition in d dimensions = d-dimensional array of wavelet coefficients
  - Coefficient *support region* = d-dimensional rectangle of cells in the original data array
  - *Sign* of coefficient's contribution can vary along the quadrants of its support

Support regions & signs for the 16 nonstandard 2-dimensional Haar coefficients of a 4X4 data array A



Multi-dimensional Haar Error Trees

- Conceptual tool for data reconstruction – more complex structure than in the 1-dimensional case
  - Internal node = *Set* of (up to) 2^d-1 coefficients (identical support regions, different quadrant signs)
  - Each internal node can have (up to) 2^d children (corresponding to the quadrants of the node's support)
- Maintains *linearity* of reconstruction for data values/range sums



Error-tree structure for 2-dimensional 4X4 example (data values omitted)

Constructing the Wavelet Decomposition

Joint Data Distribution Array

Attr2 \ Attr1	0	1	2	3
3				
2				
1		6		3
0			4	

Relation (ROLAP) Representation

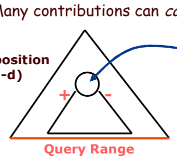
Attr1	Attr2	Count
2	0	4
1	1	6
3	1	3

- 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]

Decomposition Tree (1-d)



Contribution to range sum = 0

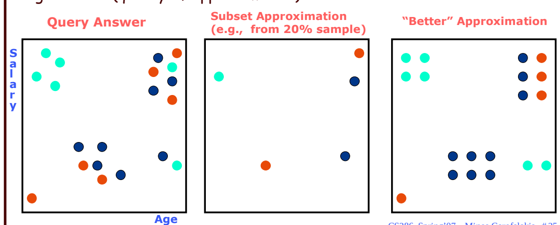
Only nodes on the path to range endpoints can have nonzero contributions (Extends naturally to multi-dimensional range sums)

Outline

- Intro & Approximate Query Answering Overview
- One-Dimensional Synopses
- Multi-Dimensional Synopses and Joins
- Set-Valued Queries
  - Error Metrics
  - Using Histograms
  - Using Samples
  - Using Wavelets
- Discussion & Comparisons
- 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



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

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 (generate approximate points, e.g., using *kernels* [BKS99, GKT00])
  - 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