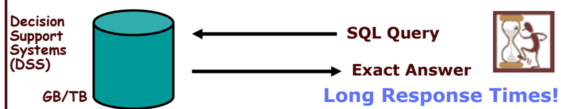
Approximation Techniques for Data Management Systems

"We are drowning in data but starved for knowledge"

John Naisbitt

Traditional Query Processing



- 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?"

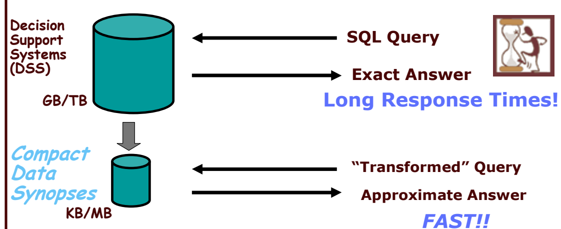
Fast Approximate Answers

- 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 (with 95% confidence) in 10 seconds
> vs. $59,152.25 in 10 minutes

- Achieved by answering the query based on *compact synopses* of the data
- Speed-up obtained because synopses are orders of magnitude smaller than the original data

Approximate Query Processing



- How do you *build effective data synopses???*

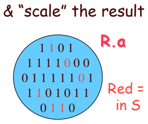
Sampling: Basics

- 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 * count(*) from S where S.a = 0

    Est. count = 5*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 ± 600 with prob ≥ 90%

Sampling: Confidence Intervals

Method	90% Confidence Interval (±)	Guarantees?
Central Limit Theorem	1.65 * σ(S) / sqrt(|S|)	as |S| → ∞
Hoeffding	1.22 * (MAX-MIN) / sqrt(|S|)	always
Chebyshev (known σ(R))	3.16 * σ(R) / sqrt(|S|)	always
Chebyshev (est. σ(R))	3.16 * σ(S) / sqrt(|S|)	as σ(S) → σ(R)

Confidence intervals for Average: select avg(R.A) from R

(Can replace R.A with any arithmetic expression on the attributes in R)
σ(R) = standard deviation of the values of R.A; σ(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 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

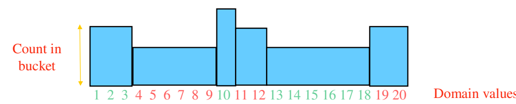
One-Pass Uniform Sampling

- 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

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
- Recently explored as a tool for fast approximate query processing

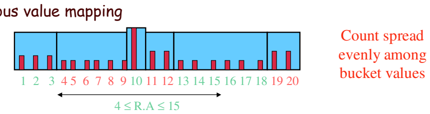
1-D Histograms



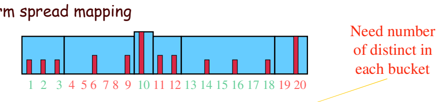
- Number of buckets B << domain size
- Each bucket just stores a *total count*
  - Distributed uniformly across values in the bucket
- *Partition criteria*
  - *Equi-width*: equal number of domain values per bucket (bad!!)
  - *Equi-depth/height*: equal count ("mass") per bucket
  - *V-Optimal*: minimize total variance of value counts in buckets

Answering Queries Using 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
- Inside each bucket: ***Uniformity Assumption***
  - Continuous value mapping



  - Uniform spread mapping



Haar Wavelet Synopses

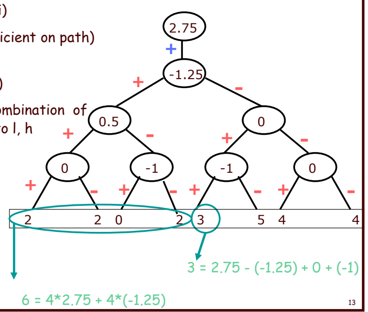
- Wavelets: mathematical tool for hierarchical decomposition of functions/signals
- Haar wavelets: simplest wavelet basis, easy to understand and implement
  - *Recursive pairwise averaging and differencing* at different resolutions

Resolution	Averages	Detail Coefficients
3	D = [2, 2, 0, 2, 3, 5, 4, 4]	----
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[-1.25]

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***)
  - Conceptual tool to "visualize" *coefficient supports* & *data reconstruction*
- Reconstruct data values d(i)
  - d(i) = $\sum$(+/-1) * (coefficient on path)
- Range sum calculation d(l:h)
  - d(l:h) = simple linear combination of coefficients on paths to l, h
- Only O(logN) terms



13

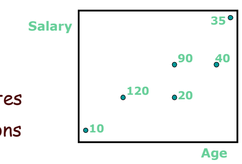
Wavelet Data Synopses

- Compute Haar wavelet decomposition of D
- *Coefficient thresholding*: only B<<|D| coefficients can be kept
  - B is determined by the available synopsis space
- Approximate query engine can do all its processing over such compact coefficient synopses (joins, aggregates, selections, etc.)
- Conventional thresholding: Take B largest coefficients in *absolute normalized value*
  - Normalized Haar basis: divide coefficients at resolution j by $\sqrt{2^j}$
  - All other coefficients are ignored (assumed to be zero)
  - *Provably optimal* in terms of the overall Sum-Squared (L2) Error

14

Multi-dimensional Data Synopses

- Problem: Approximate the *joint data distribution* of multiple attributes
  - Motivation
    - Selectivity estimation for queries with multiple predicates
    - Approximating general relations

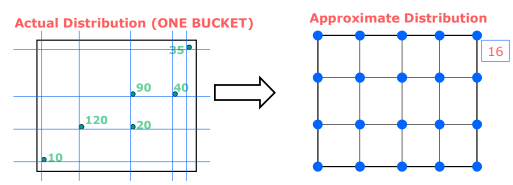


- Conventional approach: Attribute-Value Independence (AVI) assumption
  - sel(p(A1) & p(A2) & . . .) = sel(p(A1)) * sel(p(A2) * . . .
  - Simple -- one-dimensional marginals suffice
  - BUT: almost always inaccurate, gross errors in practice

15

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) = no. of distinct values along Ai, F = total bucket frequency
  - approximate data points on a n(1)*n(2)*. . . uniform grid, each with frequency F / (n(1)*n(2)*. . .)



16

Data Synopses in Commercial DBMSs

- Sampling operators ans 1-D histograms are available in most commercial DBMSs
  - Oracle, DB2, SQL Server,…
  - Used internally but also exposed to user (e.g., store "sample view")
  - SQL Server has support for 2-D histograms!
- The next step: *Synopses for XML!?!*
  - How do you effectively summarize a graph structure for queries like "//a//b[d]/*/c" ??

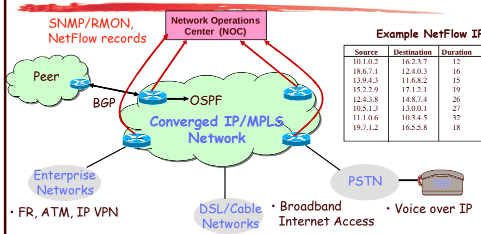
17

Data-Stream Management

- Traditional DBMS – data stored in finite, persistent data sets
- Data Streams – distributed, continuous, unbounded, rapid, time varying, noisy, . . .
- Data-Stream Management – variety of modern applications
  - Network monitoring and traffic engineering
  - Telecom call-detail records
  - Network security
  - Financial applications
  - Sensor networks
  - Web logs and clickstreams

18

Networks Generate Massive Data Streams



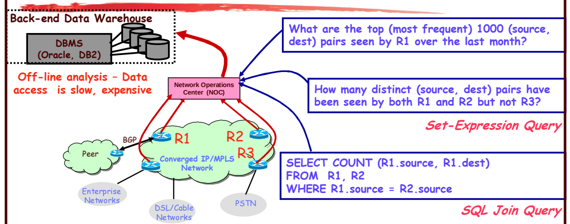
Example NetFlow IP Session Data

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

- SNMP/RMON/NetFlow data records arrive 24x7 from different parts of the network
- Truly massive streams arriving at rapid rates
  - AT&T collects 600-800 GigaBytes of NetFlow data each day!
- Typically shipped to a back-end data warehouse (off site) for off-line analysis

19

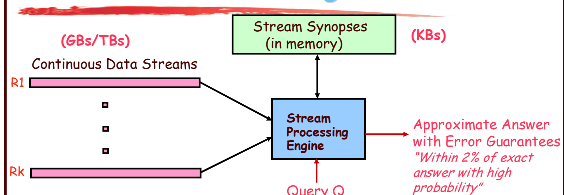
Real-Time Data-Stream Analysis



- Need ability to process/analyze network-data streams *in real-time*
  - As records stream in: look at records *only once in arrival order!*
  - Within resource (CPU, memory) limitations of the NOC
- Critical to important NM tasks
  - Detect and react to Fraud, Denial-of-Service attacks, SLA violations
  - Real-time traffic engineering to improve load-balancing and utilization

20

Data-Stream Processing Model



- Approximate answers often suffice, e.g., trend analysis, anomaly detection
- Requirements for stream synopses
  - *Single Pass:* Each record is examined at most once, in (fixed) arrival order
  - *Small Space:* Log or polylog in data stream size
  - *Real-time:* Per-record processing time (to maintain synopses) must be low

21

Distinct Value Estimation

- Problem: Find the *number of distinct values* in a stream of values with domain [0,...,N-1]
  - Zeroth frequency moment F_0, L0 (Hamming) stream norm
  - Statistics: number of *species or classes* in a population
  - Important for query optimizers
  - *Network monitoring:* distinct destination IP addresses, source/destination pairs, requested URLs, etc.
- Example (N=64) Data stream: 3 0 5 3 0 1 7 5 1 0 3 7

  Number of distinct values: 5
- Hard problem for random sampling! [CCMN00]
  - Must sample almost the entire table to guarantee the estimate is within a factor of 10 with probability > 1/2, regardless of the estimator used!

22

Hash (aka FM) Sketches for Count Distinct

- Assume a hash function h(x) that maps incoming values x in [0,..., N-1] *uniformly* across [0,..., 2^L-1], where L = O(logN)
- Let lsb(y) denote the position of the least-significant 1 bit in the binary representation of y
  - A value x is mapped to lsb(h(x))
- Maintain *Hash Sketch* = BITMAP array of L bits, initialized to 0
  - For each incoming value x, set BITMAP[lsb(h(x))] = 1

x = 5 → h(x) = 101100 → lsb(h(x)) = 2

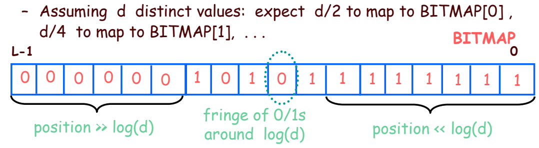
BITMAP

5	4	3	2	1	0
0	0	0	1	0	0

23

Hash (aka FM) Sketches for Count Distinct

- By uniformity through h(x): Prob[BITMAP[k]=1] = Prob[10^k] = $\frac{1}{2^{k+1}}$
  - Assuming d distinct values: expect d/2 to map to BITMAP[0] , d/4 to map to BITMAP[1], . . .



- Let R = position of rightmost zero in BITMAP
  - Use as indicator of log(d)
- [FM85] prove that E[R] = $\log(\phi d)$, where $\phi = .7735$
  - Estimate d = $2^R/\phi$
  - Average several iid instances (different hash functions) to reduce estimator variance

24

A Little Streaming Puzzle…

- *Input:* A stream of N numbers/elements
- *Output:* The stream *majority element* (if one exists)
  - e is a majority element if frequency(e) > N/2
- Q: How do you do this in *small space*??
  - Hint: Use just *two* memory locations
  - Hint++: Look at this as a "knockout tournament"
- Feeling adventurous?
  - How do you do the same majority query over a stream of *insertions and deletions?*
  - *Input*: Stream of <e, +> = insert e , <e, -> = delete e
  - Hint: Use a little more memory…

25

In Summary: Not your parents' DBMS!

- Database/data-management research goes far beyond the basics!
- Extends from distributed systems to theory to approximation algorithms to probability/statistics to …
  - Applications: data mining, sensornets, p2p, …
  - Just pick up a copy of recent SIGMOD/VLDB proceedings
- More and more relevant in dealing with the *"data tsunami"*
  - Data is everywhere! And, it's constantly growing in volume!
- Exciting, relevant research!

26

More details…

•Tutorial slides on approximate query processing and data streams

http://www2.berkeley.intel-research.net/~minos/tutorials.html

27