CSCI585

Wavelet-Based OLAP Approaches

C. Shahabi

Motivation: New Multidimensional Data Intensive Applications

- **Multidimensional data sets:** (w/ dimension & measure)
  - Remote sensory date (from JPL):
    <*latitude, longitude, altitude, time, temperature*>
  - Sensor readings from GPS ground stations (from NASA):
    <*lat, long, t, velocity*>
  - Petroleum sales (from Digital-Government research center):
    <*location, product, year, month, volume*>
  - ACOUSTIC data (from UCLA sensor-network project):
    <*IPAQ-id, volume-id, event#, time, value*>
  - Market data (from NCR): <*store-location, product-id, date, price, sale*>
- Large size, e.g., current (toy!) NASA/JPL data set:
  - Past 10 years, sampling twice a day, at a lat-long-alt grid of 64 * 128 * 16, recording 8 bytes of temperature & 16 bytes of dimensions
  - This is 6 MB of data per day; a total of 21 GB for 10 years
  - Increase: twice an hour sampling, 1024 * 4096 * 128 grid, …

Motivation: Multidimensional Applications

- I/O and computationally complex queries
  - Range-aggregate queries (w/ *aggregate function*)
    - Average temperature, given an area and time interval
    - Average velocity of upward movement of the station
    - Total petroleum sales volume of a given product in a given location and year
    - Number of jackets sold in Seattle in Sep. 2001
  - Tougher queries:
    - Covariance of temperature and altitude (correlation)
    - Variance of sale of petroleum in 2002 in CA
- Quick response-time (interactive):
  - the results can be approximate and/or progressively become exact

Recap!

- Multidimensional data
- Large data
- Aggregate queries
- Approximate answers
- Progressive answers

- Multi-resolution compression

- Wavelets!

Data Cube Approximation and Histograms via Wavelets

J. S. Vitter, M. Wang and B. Iyer

Outline

- Motivation
- Technique
- Complexity Analysis

Motivation

- There are a number of scenarios in which a user may prefer an approximate answer in a few seconds over an exact answer that requires tens of minutes or more to compute.
- Another consideration is that the data cube may be remote and currently unavailable, so that finding an exact answer is not an option, until the data again become available.

Technique

1. In a preprocessing step, they form the partial sum data cube P from the (raw) data cube A. (In their method, they further process P by replacing each cell value by its natural logarithm.)
2. They compute the wavelet decomposition of P, obtaining a set of N coefficients, where N is the size of array A.
3. They keep only the C most significant wavelet coefficients, for some C that corresponds to the desired storage usage and accuracy. The choice of which C coefficients to keep depends upon the particular thresholding method they use.

1. Computing the Partial Sum Datacube

- Why wavelet decompose P and not A?
  - P is monotone nondecreasing, and a compact data cube built on P seems to give a better approximation than one built directly on A.
  - To answer a range-sum query using the compact data cube built on P, all they need to do in the on-line phase is to reconstruct the values corresponding to the boundaries of the ranges (instead of reconstructing all the values covered by the query, as in an extended data cube).

2. Wavelet Decomposition of the Partial Sum Data Cube P

- Wavelet basis function: Haar wavelets:
  - h=[½ ½]
  - g=[-½ ½]
- One dimensional "signal": [2, 2, 7, 11]

Resolution	Averages	Detail Coefficients
4	[2, 2, 7, 11]	
2	[2, 9]	[0, 2]
1	[5½]	[3½]

- Wavelet transform of the original signal is the overall average of the original signal followed by the detail coefficients in the order of increasing resolutions: [5½ , 3½ , 0, 2]

2. Wavelet Decomposition of the Partial Sum Data Cube P

- **Loss less: the original signal can be reconstructed**
- **If signal correlated, lots of zeros in details**
- **General: other filters, convolution, down-sampling**
- **Multidimensional transform:**
  - **Perform a series of one-dimensional decompositions.**
  - **For example, in the two-dimensional case, we first apply the one-dimensional wavelet transform to each row of the data. Next, we treat these transformed rows as if they were themselves the original data, and we apply the one-dimensional transform to each column.**

3. Thresholding and Error Measures

- Keeping only C << N coefficients
- Question: Which are the "best" C coefficients to keep, so as to minimize the error of approximation
- It is well-known that thresholding by choosing the C largest (in absolute value) wavelet coefficients after normalization is provably optimal in minimizing the 2-norm (Euclidian distance) of the absolute errors, among all possible choices of C nonzero coefficients, assuming that the wavelet basis functions are orthonormal.
- The C wavelet coefficients together with their C indices (in the one-dimensional order of cells), form the compact data cube.

Complexity Analysis

- Storage: 2C
- Off-line Transformation of P:

  $O(N/B \ \log_{M/B} N/B)$ where B is the disk block size, N is the total cube size, and M is the internal memory size
- Partial sum value for a given cell can be computed using O(dC) space in

  $O(\Sigma_{1<=i<=d} \min\{C, \log|D_i|\})$ where $|D_i|$ is the size of dimension i of the cube (out of d dimensions)

ProPolyne: A Fast Wavelet-based Algorithm for Progressive Evaluation of Polynomial Range-Sum Queries

Rolfe Schmidt and Cyrus Shahabi
University of Southern California
Dept. of Computer Science
Los Angeles, CA 90089-0781
shahabi@usc.edu
http://infolab.usc.edu

Approach: Enabling Data Manipulation, Query & Analysis in the WAVELET Domain

- Everybody else's idea: let's compress data
  - Reason: save space? No not really!
  - Implicit reason: queries deal with smaller data sets and hence faster (not always true!)
  - More problems: not only query results can never be 100% accurate anymore, but also different queries can have very different error rates given their areas of interest
  - Why? At the data population time, we don't know which coefficients are more/less important to our queries! (also observed by *[Garofalakis & Gibbons, SIGMOD'02],* but they proposed other ways to drop coefficients assuming a uniform workload)
    - Different than the signal-processing objective to reconstruct the entire signal as good as possible

Approach: Enabling Data Manipulation, Query & Analysis in the WAVELET Domain

- Our idea/distinction: storage is cheap and queries are ad-hoc; let's keep all the wavelet coefficients! (no data compression)
- Opportunity: At the query time, however, we have the knowledge of what is important to the pending query
- ProPolyne: Progressive Evaluation of Polynomial Range-Aggregate ***Query***

Outline

- ProPolyne: Overview and Features
- ProPolyne: Details
- Comparison Table
- Performance Results
- Conclusion
- How to Evaluate Multiple Range-Sum Queries Progressively

Overview of ProPolyne

- Define range-sum query as dot product of *query vector* and *data vector*
- Offline: Multidimensional wavelet transform of data
- At the query time: *"lazy"* wavelet transform of query vector (very fast)
- Dot product of query and data vectors in the transformed domain → exact result in $O(2 \log N)^d$
- Choose high-energy query coefficients only → fast approximate result (90% accuracy by retrieving < 10% of data)
- Choose query coefficients in order of energy → progressive result

ProPolyne Features

- "Measure" can be any polynomial on any combination of attributes
  - Can support COUNT, SUM, AVERAGE
  - *Also* supports Covariance, Kurtosis, etc.
  - All using *one* set of pre-computed aggregates
- Independent from how well the data set can be compressed/approximated by wavelets
  - Because: We show "range-sum queries" can always be approximated well by wavelets (not always HAAR though!)
- Low update cost: O(log^d N)
- Can be used for *exact, approximate* and *progressive* range-sum query evaluation

Outline

- ProPolyne: Overview and Features
- ProPolyne: Details
  - Polynomial Range-Sum Queries as Vector Queries
  - Naive Evaluation of Vector Queries
  - Fast Evaluation of Vector Queries
  - Progressive/Approximate Evaluation of Vector Queries
- Comparison Table
- Performance Results
- Conclusion
- How to Evaluate Multiple Range-Sum Queries Progressively

Polynomial Range-Sum Queries

- Polynomial range-sum queries: *Q(R,f,I)*
  - *I* is a finite instance of schema *F*
  - *R* *SubSetOf* Dom(*F*), is the range
  - *f* : Dom(*F*) → R is a polynomial of degree δ

$$Q(R,f,I) = \sum_{\bar{x} \in I \cap R} f(\bar{x})$$

- Example: F = (Age, Salary)
- R: (25 < age < 40) & (55k < salary < 150k)

Age	Salary
25	$50k
28	$55k
30	$58k
50	$100k
55	$130k
57	$120k

$\Big\}\ I$

$$COUNT\ :\ f(\bar{x}) \equiv 1(\bar{x}) = 1$$

$$Q(R,1,I) = \sum_{\bar{x} \in R \cap I} 1(\bar{x}) = 1(28,55K) + 1(30,58k) = 2$$

$$SUM\ :\ f(\bar{x}) \equiv salary(\bar{x})$$

$$Q(R, salary, I) = \sum_{\bar{x} \in R \cap I} f(\bar{x}) = salary(28,55K) + salary(30,58k) = 113k$$

$$Q(R, salary \times age, I) = \sum_{\bar{x} \in R \cap I} salary(\bar{x}) \times age(\bar{x}) = f(28,55K) + f(30,58k) = 3280M$$

$$Cov(age, salary) = \frac{Q(R, salary \times age, I)}{Q(R,1,I)} - \frac{Q(R, age, I)Q(R, salary, I)}{(Q(R,1,I))^{\wedge 2}}$$

Polynomial Range-Sum Queries as "Vector Queries"

- The data frequency distribution of I is the function Δ_I : Dom(F) → Z that maps a point x to the number of times it occurs in I
- To emphasize the fact that a query is an operator on the data frequency distribution, we write

$$Q(R, f, I) = Q(R, f, \Delta_I)$$

- Example: Δ(25,50)=Δ(28,55)=…=Δ(57,120)=1 and Δ(x)=0 otherwise.

Hence: $$Q(R, f, \Delta_I) = \sum_{\bar{x} \in Dom(F)} f(\bar{x}) \chi_R(\bar{x}) \Delta_I(\bar{x})$$

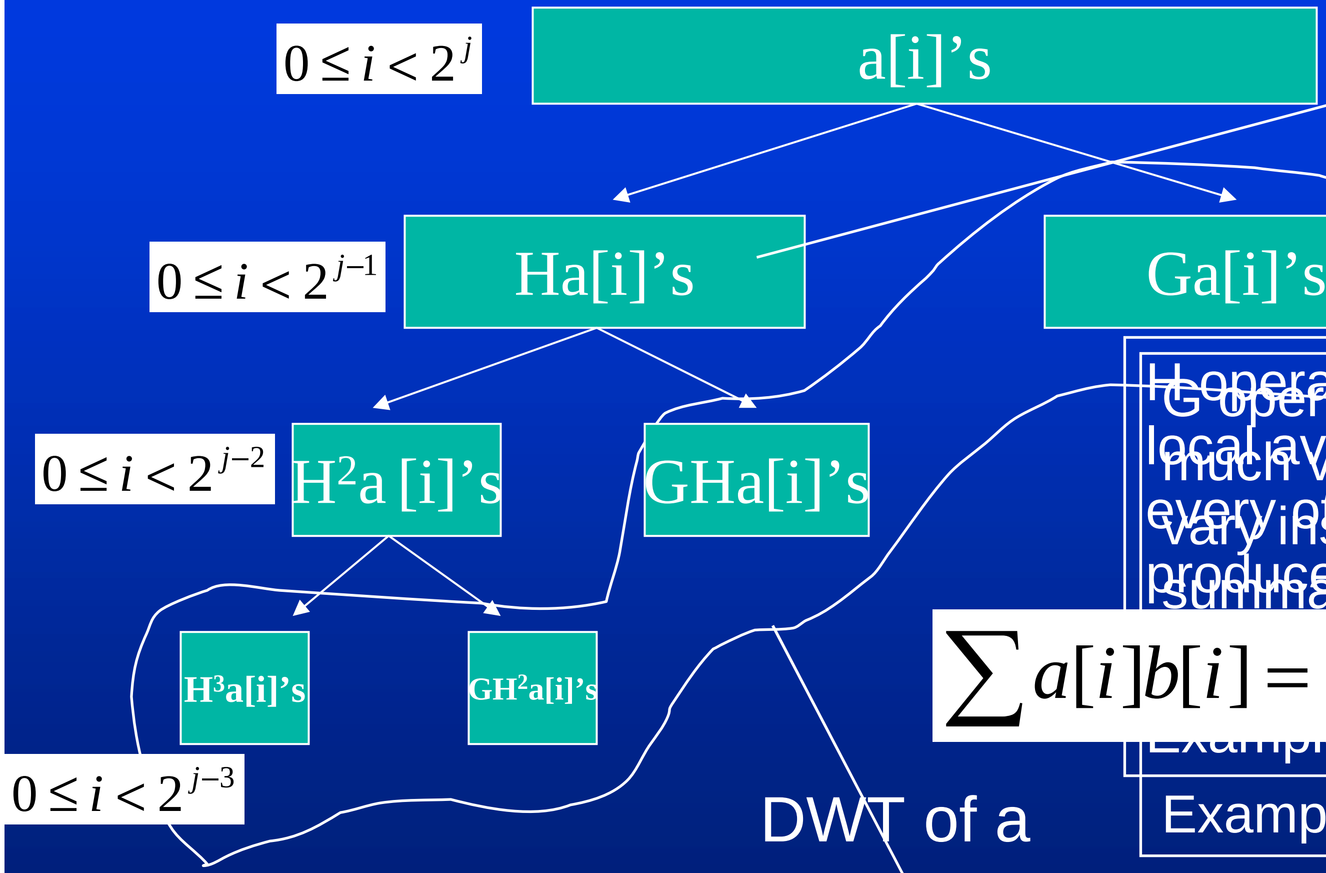
where: $$\begin{cases} \chi_R(\bar{x}) = 1 & \text{if } \bar{x} \in R \\ \chi_R(\bar{x}) = 0 & \text{if } \bar{x} \notin R \end{cases}$$

Age	Salary
25	$50k
28	$55k
30	$58k
50	$100k
55	$130k
57	$120k

I

Or: $$\underbrace{Q(R, f, \Delta_I)}_{\text{Vector Query}} = \langle \underbrace{f\chi_R}_{\text{query}}, \underbrace{\Delta_I}_{\text{data}} \rangle$$

Overview of Wavelets



Summary coefficients of ***a*** at level 2

Detail coefficients of ***a*** at level 2

H operator: computes a local average of array ***a*** at every other point to produce an array of ... ***Ha***

Example (Haar) h=[1/2,1/2]

G operator: measures how much values in the array ***a*** vary inside each of the summarized blocks to ... detail

Example (Haar) g=[1/2,-1/2]

$$\sum a[i]b[i] = \sum \hat{a}[\eta]\hat{b}[\eta]$$

DWT of a

$\hat{a}$ aka wavelet coefficients of ***a***

Naive Evaluation of Vector Queries Using Wavelets

- Hence, vector queries can be computed in the wavelet-transformed space as:

$$Q(R, f, \Delta) = (f\hat{\chi}_R, \hat{\Delta}) = \sum_{\eta_0,\ldots,\eta_{d-1}=0}^{N-1} f\hat{\chi}_R(\eta_0,\ldots,\eta_{d-1})\hat{\Delta}(\eta_0,\ldots,\eta_{d-1})$$

- Algorithm:
  - Off-line transformation of data vector (or "data distribution function", i.e., Δ, to be exact)
    - O (|I|$\lceil^d$*log*^{d}N) for sparse data, O (|I|) = N^d for dense data
  - Transform the query vector at submission
    - O (*N*^{d}) !
  - Sum-up the products of the corresponding elements of data and query vectors
    - Retrieving elements of data vector: O (N^d) !

Fast Evaluation of Vector Queries Using Wavelets

- **Main intuitions:**
  - **“query vector” can be transformed quickly because most of the coefficients are known in advance**
  - **“Transformed query vector” has a large number of negligible (e.g., zero) values (independent on how well data can be approximated by wavelet)**
  - **Example: Haar filter & COUNT function on R=[5,12] on the domain of integers from 0 to 15:**

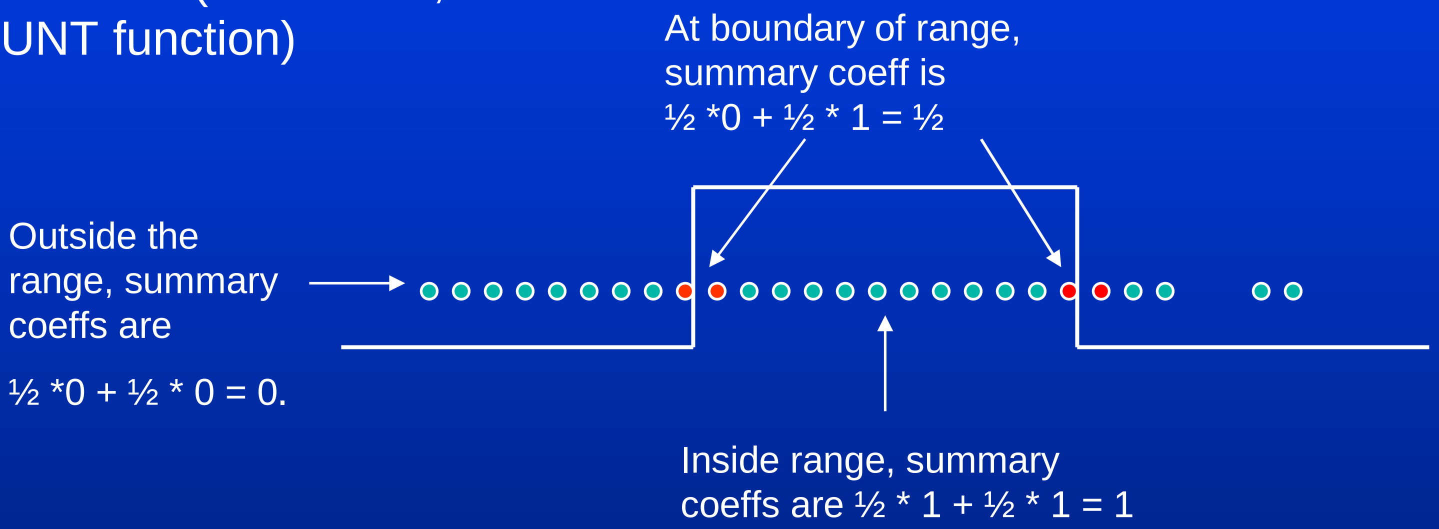
$$\chi_R = \{0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0\}$$

$$\hat{\chi}_R = \{\underbrace{2}_{H^4a}, \underbrace{\frac{1}{2}}_{GH^3a}, \underbrace{\frac{3}{2\sqrt{2}}, -\frac{3}{2\sqrt{2}}}_{GH^2a}, \underbrace{0, \frac{1}{2}, 0, -\frac{1}{2}}_{GHa}, \underbrace{0, 0, \frac{1}{\sqrt{2}}, 0, 0, 0, -\frac{1}{\sqrt{2}}, 0}_{Ga}\}$$

At each step, you know the zeros

The Lazy Wavelet Transform

Computing Summary Coefficients (Haar Filter, COUNT function)


At boundary of range, summary coeff is ½ *0 + ½ * 1 = ½
Outside the range, summary coeffs are
½ *0 + ½ * 0 = 0.
Inside range, summary coeffs are ½ * 1 + ½ * 1 = 1

All summary coefficients computed in CONSTANT time!

Summary coefficient array looks almost exactly like original array.

The only "interesting" activity happens on the boundary.

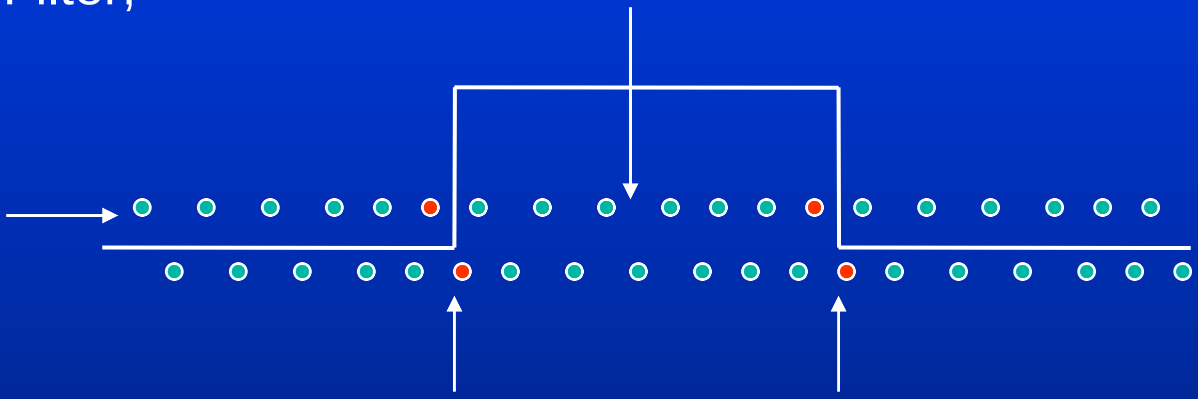
The Lazy Wavelet Transform

Computing Detail Coefficients (Haar Filter, COUNT function)

Outside the range, detail coeffs are

½ *0 - ½ * 0 = 0.

Inside range, detail coeffs are ½ * 1 - ½ * 1 = 0



At lower boundary of range, detail coeff is ½ *0 - ½ * 1 = -½

At upper boundary of range, detail coeff is ½ *1 - ½ * 0 = ½

All detail coefficients computed in CONSTANT time!

All but 2 detail coefficients at each level are equal to zero!

The only “interesting” activity happens on the boundary.

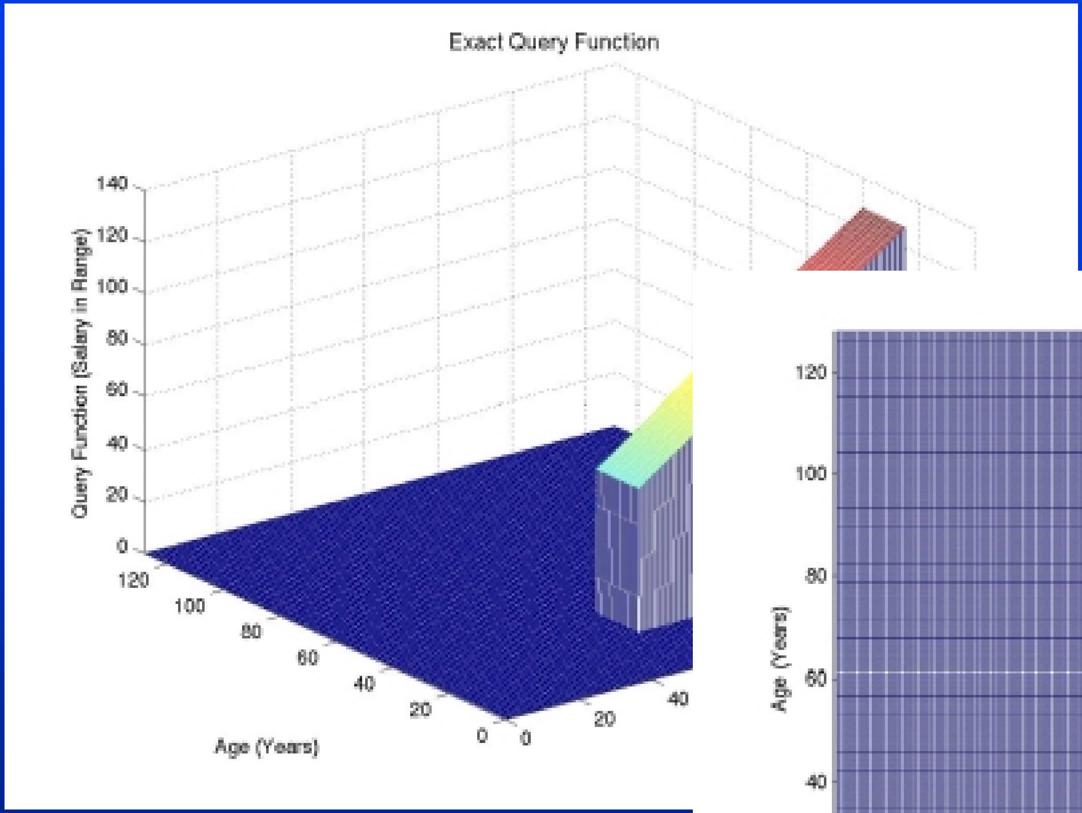
Fast Evaluation of Vector Queries Using Wavelets …

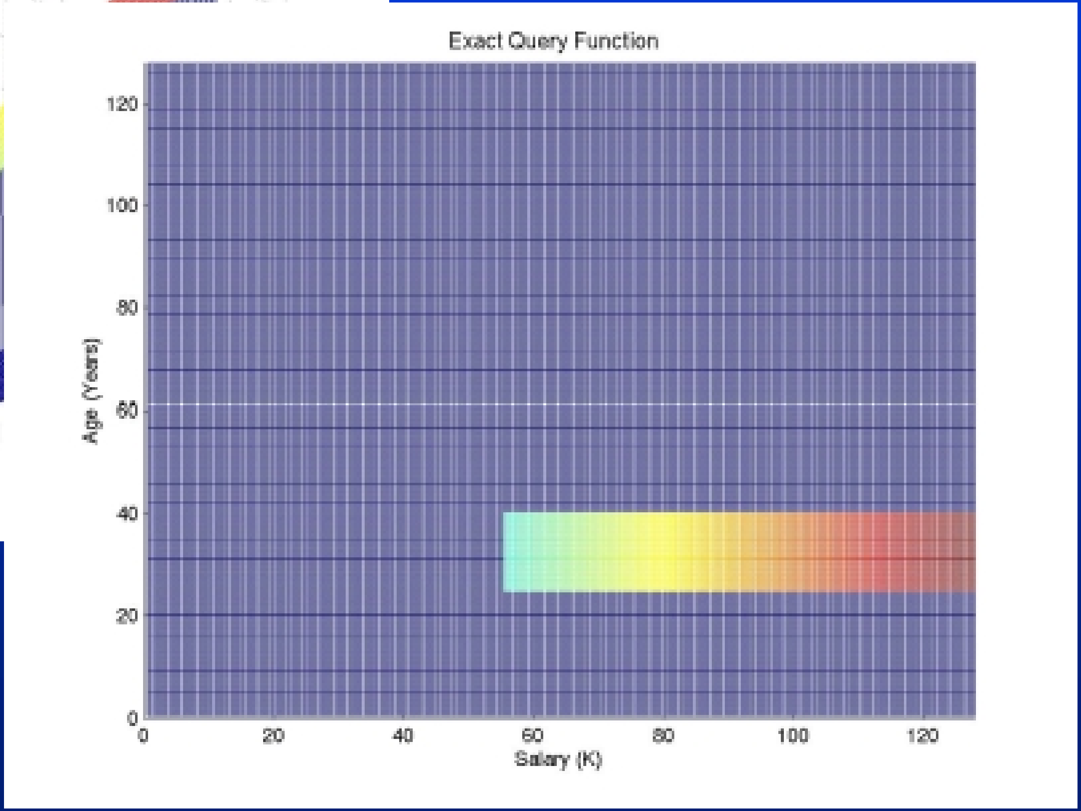
- Technical Requirements:
  - Wavelets should have small support (i.e., the shorter the filter, the better)
  - Wavelets must satisfy a “moment condition”
  - Supports any Polynomial Range-Sum up to a degree determined by the choice of wavelets
    - E.g., Haar can only support degree 0 (e.g., COUNT), while db4 can support up to degree 1 (e.g., SUM), and db6 for degree 2 (e.g., VARIANCE)
- Standard DWT: O (N)
- Our lazy wavelet transform: O (l *log* N),

  where l is the length of the filter

CSCI585

Exact Evaluation of Vector Queries



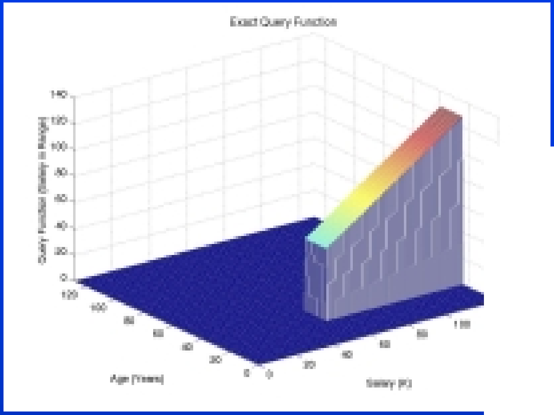


Query:
SUM(salary) when
(25 < age < 40) &
(55k < salary < 150k)

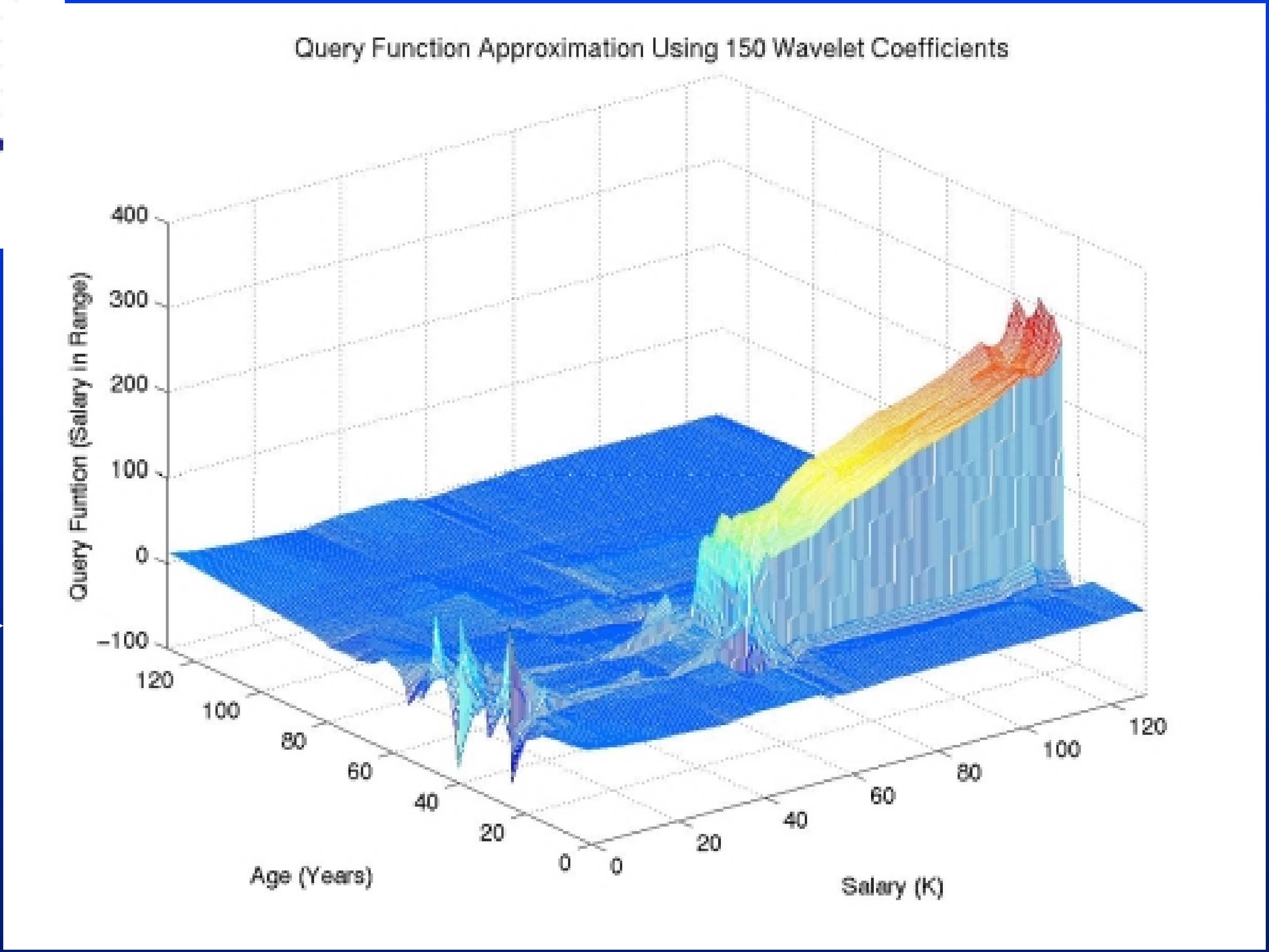
# of Nonzero Coordinates: 4380

# of Wavelet Coefficients: 837

Approximate Evaluation of Vector Queries

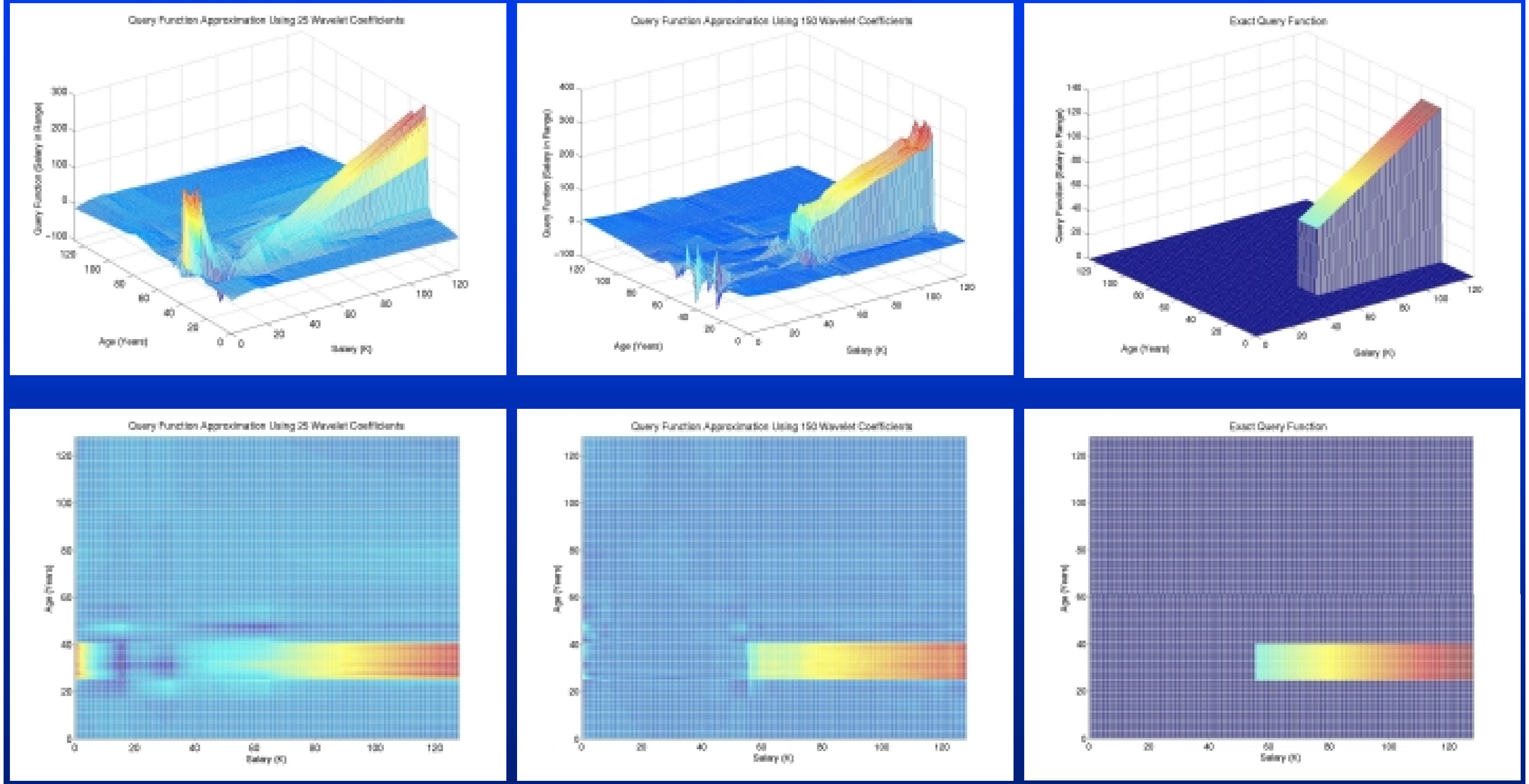


With 150 coefficients:
It is as if this query is being Submitted or evaluated



CSCI585

Progressive Evaluation of Vector Queries



C. Shahabi

Name of Technology	Research Group	Query Cost	Update Cost	Storage Cost	Aggregate Function Support	Query Evaluation Support	Measure Known at Population?
PROPOLYNE 2001	USC Schmidt & Shahabi	$\lg^{d} N(4\delta)^{d}$	$\lg^{d} N(2\delta)^{d}$	N^{d}	Polynomial Range-Sums of degree δ	Exact, Approximate, Progressive	No
PROPOLYNE-FM 2001	USC Schmidt & Shahabi	$2^{d} \lg^{d-1} N$	$\lg^{d-1} N$	N^{d-1}	COUNT and SUM	Exact, Approximate, Progressive	Yes
Space-Efficient Dynamic Data Cube 2000	UCSB El-Abbadi & Agrawal et. al	$2^{d} \lg^{d-1} N$	$\lg^{d-1} N$	N^{d-1}	COUNT and SUM	Exact	Yes
Relative Prefix-Sum 1999	UCSB	4^{d-1}	$N^{(d-1)/2}$	N^{d-1}	COUNT and SUM	Exact	Yes
Prefix-Sum 1997	IBM Agrawal et. al	2^{d-1}	N^{d-1}	N^{d-1}	COUNT and SUM	Exact	Yes
pCube/MRATree 2000/2001	UCSB and UC Irvine (Mehrota et. al)	N^{d-1}	$\lg N$	N^{d}	COUNT and SUM	Exact, Approximate, Progressive	Yes
Compact Data Cube 1998-2000	Duke and IBM (Vitter et. al)	small	?	small	COUNT and SUM	Approximate	Yes
Optimal Histograms 2001	AT&T (Muthu et. al)	small	?	small	COUNT and SUM	Approximate	Yes
Kernel Density Estimators 1999	Microsoft (Fayyad et. al)	small	?	small	All efficiently computable functions	Approximate	No

Conclusion

- A novel pre-aggregation strategy
- Supports conventional aggregates: COUNT, SUM and beyond: multivariate statistics
- First pre-aggregation technique that does not require measures be specified a priori
  - Measures treated as functions of the attributes at the time
- Provides a data independent progressive and approximate query answering technique
- With provably poly-logarithmic worst-case query and update costs
- And storage cost comparable or better than other pre-aggregation methods