Wavelet-Based Histograms for Selectivity Estimation

Yossi Matias*
Department of Computer Science
Tel Aviv University, Israel
matias@math.tau.ac.il

Jeffrey Scott Vitter†
Department of Computer Science
Duke University
jsv@cs.duke.edu

Min Wang‡
Department of Computer Science
Duke University
minw@cs.duke.edu

Abstract

Query optimization is an integral part of relational database management systems. One important task in query optimization is selectivity estimation, that is, given a query P, we need to estimate the fraction of records in the database that satisfy P. Many commercial database systems maintain histograms to approximate the frequency distribution of values in the attributes of relations.

In this paper, we present a technique based upon a multiresolution wavelet decomposition for building histograms on the underlying data distributions, with applications to databases, statistics, and simulation. Histograms built on the cumulative data distributions give very good approximations with limited space usage. We give fast algorithms for constructing histograms and using them in an on-line fashion for selectivity estimation. Our histograms also provide quick approximate answers to OLAP queries when the exact answers are not required. Our method captures the joint distribution of multiple attributes effectively, even when the attributes are correlated. Experiments confirm that our histograms offer substantial improvements in accuracy over random sampling and other previous approaches.

*Also affiliated with Bell Laboratories, Murray Hill, NJ.

†Part of this work was done while the author was visiting Bell Laboratories in Murray Hill, NJ. Supported in part by Army Research Office MURI grant DAAH04–96–1–0013 and by National Science Foundation research grant CCR–9522047.

‡Supported in part by an IBM Graduate Fellowship and by Army Research Office MURI grant DAAH04–96–1–0013.

1 Introduction

Several important components in a database management system (DBMS) require accurate estimation of the selectivity of a given query. For example, query optimizers use it to evaluate the costs of different query execution plans and choose the preferred one.

The set of predicates we are going to consider in this paper is the set of selection queries, in particular, *range predicates* of the form $a \le X \le b$, where X is a non-negative attribute of the domain of a relation R and a and b are constants. The set of *equal predicates* is the subset of the range predicates that have $a = b$. The set of *one-side range predicates* is the special case of range predicates in which $a = -\infty$ or $b = \infty$.

We adopt the notations in [16] to describe the data distributions and various histograms. The *domain* $D = \{0, 1, 2, \ldots, N-1\}$ of an attribute X is the set of all possible values of X. The value set $V \subseteq D$ consists of the n distinct values of X that are actually present in relation R. Let $v_1 < v_2 < \cdots < v_n$ be the n values of V. The *spread* s_i of v_i is defined as $s_i = v_{i+1} - v_i$. (We take $s_0 = v_1$ and $s_n = 1$.) The *frequency* f_i of v_i is the number of tuples in which X has value v_i. The *cumulative frequency* c_i of v_i is the number of tuples $t \in R$ with $t.X \le v_i$; that is, $c_i = \sum_{j=1}^{i} f_j$. The data distribution of X is the set of pairs $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}$. The *cumulative data distribution* of X is the set of pairs $\mathcal{T}^{\mathcal{C}} = \{(v_1, c_1), (v_2, c_2), \ldots, (v_n, c_n)\}$. The *extended cumulative data distribution* of X, denoted by $\mathcal{T}^{\mathcal{C}^+}$, is the cumulative data distribution of $\mathcal{T}^{\mathcal{C}}$ extended over the entire domain D by assigning zero frequency to every value in $D - V$.

2 Previous Approaches

The goal of any histogram is to accurately approximate the underlying distribution. Several different histograms have been proposed in the literature. Poosala et al [16, 13] propose a taxonomy to capture all previously proposed histograms; new histograms types can be derived by combining effective aspects of different histogram methods. Among the histograms discussed in [16, 13], the MaxDiff(V,A) histogram gives best overall performance. MaxDiff(V,A) uses area as a "source parameter" in order to choose the bucket boundaries; area is defined for each value in the value set V as the product of the value's frequency and its spread. In a MaxDiff(V,A) histogram with β buckets, there is a bucket boundary between two source parameter values that are adjacent (in attribute value order) if the difference between these values is one of the $\beta - 1$ largest such differences. By using *area* as a source parameter, MaxDiff(V,A) histogram achieves better approximation than previous well-known methods like equidepth histograms.

Poosala et al [16] also mention other histogram techniques. For example, histograms based on minimizing the variance of the source parameter such as area have a performance similar to that of MaxDiff(V,A), but are computationally more expensive to construct. They also mention a spline-based technique, but it does not perform as well as does MaxDiff(V,A).

The main challenge for histograms on multidimensional (multi-attribute) data is to capture the correlations among different attributes. We defer discussion of the multidimensional case to Section 5.

3 Our Wavelet-Based Technique

Wavelets are a mathematical tool for hierarchical decomposition of functions. Wavelets represent a function in terms of a coarse overall shape, plus details that range from broad to narrow. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the various levels of detail of the function in a space-efficient manner.

At a high level, our histogram construction algorithm works as follows:

1. In a preprocessing step, we form the *extended cumulative data distribution* $\mathcal{T}^{\mathcal{C}^+}$ of the attribute X, from the original data or from a random sample of the original data.
2. We compute the wavelet decomposition of $\mathcal{T}^{\mathcal{C}^+}$, obtaining a set of N wavelet coefficients.
3. We keep only the m most significant wavelet coefficients, for some m that corresponds to the desired storage usage. The choice of which m coefficients we keep depends upon the particular thresholding method we use.

After the above algorithm, we obtain m wavelets coefficients. The values of these coefficients, together with their positions (indices), are stored and serve as a histogram for reconstructing the approximate data distribution in the on-line phase (query phase). To compute the estimate for the number of tuples whose X value is in the range $a \leq X \leq b$, we reconstruct the approximate values for b and $a-1$ in the extended cumulative distribution function and then subtract them.

One interesting observation we made during experiments is that the wavelet approximation is more effective for selectivity estimation, especially for range queries, if the decomposition is done on the extended cumulative data distribution as described above rather than on the raw data frequencies.

Further benefits can be obtained by quantizing the wavelet coefficients and entropy-encoding the quantized coefficients. In this paper, we restrict ourselves to choosing m complete coefficients, so as to facilitate direct comparisons with previous work.

3.1 Preprocessing

In the preprocessing step, we compute $\mathcal{T}$. The extended cumulative data distribution $\mathcal{T}^{\mathcal{C}^+}$ can be easily computed from $\mathcal{T}$. Exact computation of $\mathcal{T}$ can be done by maintaining a counter for each distinct attribute value in V. When the cardinality of V is relatively small so that $\mathcal{T}$ fits in internal memory, which is often the case for the low-dimensional data we consider, we can keep a (hash) table in memory, and $\mathcal{T}$ can be obtained in one complete scan through the relation.

When the cardinality of V is too large, multiple passes through the relation will be required to obtain $\mathcal{T}$, resulting in possibly excessive I/O cost. We can instead use an I/O-efficient external merge sort to compute $\mathcal{T}$ and minimize the I/O cost [25]. The merge sort process here is different from the traditional one: During the merging process, records with the same attribute value can be combined by summing the frequencies. After a small number of multi-way passes in the merge sort, the lengths of the runs will stop increasing; the length of each run is bounded by the cardinality of V, whose size, although too large to fit in memory, is typically small in comparison with the relation size.

If for any reason it's necessary to further reduce the I/O and CPU costs of the precomputation of $\mathcal{T}$, a well-known approach is to use random sampling [12, 11]. The idea is to sample s tuples from the relation randomly and compute $\mathcal{T}$ for the sample. The sample data distribution is then used as an estimate of the real data distribution. To obtain the random sample in a single linear pass, the method of choice is the skip-based method [24] when the number of tuples T is known beforehand or the reservoir sampling variant [23] when T is unknown. A running up-to-date sample can be kept using a backing sample approach [4]. We do not consider in this paper the issues dealing with sample size and the errors caused by sampling. Our experiments confirm that wavelet-based histograms that use random sampling as a preprocessing step give estimates that are almost as good as those from wavelet-based histograms that are built on the full data. On the other hand, as we shall see in Section 6, the wavelet-based histograms (whether they use random sampling in their preprocessing or not) perform significantly better at estimation than do naive techniques based on random sampling alone.

In this paper, we do not consider the I/O complexities of the wavelet decomposition and of thresholding, since both of them are performed on the cumulative data distribution $\mathcal{T}^{\mathcal{C}^+}$ and the size of $\mathcal{T}^{\mathcal{C}^+}$ is generally not large and can fit in internal memory for the low-dimensional data we considered in this paper. For high-dimensional data, which is generally larger, the I/O efficiencies of the wavelet decomposition and of thresholding are considered in [26].

3.2 Wavelet Decomposition

The goal of the wavelet decomposition step is to represent the extended cumulative data distribution $\mathcal{T}^{\mathcal{C}^+}$ at hierarchical levels of detail.

First we need to choose wavelet basis functions. Haar wavelets are conceptually the simplest wavelet basis functions, and for purposes of exposition in this paper, we focus our discussion on Haar wavelets. They are fastest to compute and easiest to implement. We also implement a decomposition based upon linear wavelets that gives better estimation.

To illustrate how Haar wavelets work, we start with a simple example. A detailed treatment of wavelets can be found in any standard reference on the subject (e.g., [8, 20]). Suppose that the data distribution $\mathcal{T}$ of attribute X is $\{(0,2),(2,5),(3,2)\}$. We can easily derive the cumulated values $\mathcal{T}^{\mathcal{C}^+}=\{(0,2),(1,2),(2,7),(3,9)\}$. We perform a wavelet transform on the one-dimensional "signal" of the extended cumulative frequencies:

$$S = [2,\ 2,\ 7,\ 9].$$

We first average the cumulative frequencies, pairwise, to get the new lower resolution signal with values

$$[2,\ 8].$$

That is, the first two values in the original signal (2 and 2) average to 2, and the second two values 7 and 9 average to 8. Clearly, some information is lost in this averaging process. To recover the original signal from the two averaged values, we need to store some *detail coefficients*, which capture the missing information. Haar wavelets store the pairwise differences of the original values. It is easy to see that the original values can be recovered from the averages and differences.

We have succeeded in decomposing the original signal into a lower resolution (two-value) version and a pair of detail coefficients. By repeating this process recursively on the averages, we get the full decomposition:

Resolution	Averages	Detail Coefficients
4	[2, 2, 7, 9]	
2	[2, 8]	[0, 2]
1	[5]	[6]

We define the *wavelet transform* (also called *wavelet decomposition*) of the original four-value signal to be the single coefficient representing the overall average of the original signal, followed by the detail coefficients in the order of increasing resolution. Thus, for the one-dimensional Haar basis, the wavelet transform of our original cumulative frequencies is given by

$$\widehat{S} = [5,\ 6,\ 0,\ 2].$$

The individual entries are called the *wavelet coefficients*. The wavelet decomposition is very efficient computationally, requiring only $O(N)$ time to compute for a signal of N frequencies.

No information has been gained or lost by this process. The original signal has four values, and so does the transform. The original signal S can be reconstructed from $\widehat{S}$ by the following formulas:

$$S(0) = \widehat{S}(0) - \frac{1}{2}\widehat{S}(1) - \frac{1}{2}\widehat{S}(2) \tag{1}$$

$$S(1) = \widehat{S}(0) - \frac{1}{2}\widehat{S}(1) + \frac{1}{2}\widehat{S}(2) \tag{2}$$

$$S(2) = \widehat{S}(0) + \frac{1}{2}\widehat{S}(1) - \frac{1}{2}\widehat{S}(3) \tag{3}$$

$$S(3) = \widehat{S}(0) + \frac{1}{2}\widehat{S}(1) + \frac{1}{2}\widehat{S}(3) \tag{4}$$

One advantage of the wavelet transform is that in many cases a large number of the detail coefficients turn out to be very small in magnitude. Truncating these small coefficients from the representation introduces only small errors in the reconstructed signal. We can approximate the original data distribution effectively by keeping only the most significant coefficients, determined by some thresholding method, as discussed in the next subsection.

A better higher-order approximation for purposes of range query selectivity, for example, can be obtained by using linear wavelets as a basis rather than Haar wavelets. Linear wavelets share the important properties of Haar wavelets that we exploit for efficient processing. It is natural in conventional histograms to interpolate the values of items within a bucket in a uniform manner. Such an approximation corresponds to a linear function between the endpoints of the bucket. The approximation induced when we use linear wavelets is a piecewise linear function, which implies exactly this sort of linear interpolation. It therefore makes sense intuitively that the use of linear wavelets, in which we optimize directly for the best set of interpolating segments, will perform better than standard histogram techniques. For reasons of brevity, we defer further discussion to the full paper.

3.3 Thresholding

Given the storage limitation for the histogram, we can only "keep" a certain number of the N wavelet coefficients. Let m denote the number of wavelet coefficients that we have room to keep; the remaining wavelet coefficients will be implicitly set to 0. Typically we have $m \ll N$. The goal of thresholding is to determine which are the "best" m coefficients to keep, so as to minimize the error of approximation.

We can measure the error of approximation made by histograms in several ways. Let S_i be the actual size of a query q_i and let $S_i{}'$ be the estimated size of the query. We use the following three different error measures for the error e_i of query q_i:

1. The *absolute error* of a query:

$$e_i^{\text{abs}} = |S_i - S_i{}'|.$$

2. The *relative error* of a query:

$$e_i^{\text{rel}} = \frac{e_i^{\text{abs}}}{S_i} = \frac{|S_i - S_i{}'|}{S_i}, \qquad \text{for } S_i > 0.$$

3. The *combined error* of a query:

$$e_i^{\text{comb}} = \min\{\alpha \times e_i^{\text{abs}},\, \beta \times e_i^{\text{rel}}\},$$

where α and β are positive constants. (If $S_i = 0$, then we set $e_i^{\text{comb}} = \alpha \times e_i^{\text{abs}}$.)

The combined error reflects the importance of having either a good relative error or a good absolute error for each estimation. For example, for very small frequencies, it may be good enough if the absolute error is small even if the relative error is large, and for large frequencies, the absolute error may not be as meaningful as the relative error.

Once we choose which of the above measures to use in order to represent the errors of individual queries, we need to choose a norm by which to measure the error of a collection of queries. Let $e = (e_1, e_2, \ldots, e_Q)$ be the vector of errors over a sequence of Q queries. We assume that one of the above three error measures is used for each of the individual query errors e_i. For example, for absolute error, we can write $e = (e_1, e_2, \ldots, e_Q) = e^{\text{abs}} = (e_1^{\text{abs}}, e_2^{\text{abs}}, \ldots, e_Q^{\text{abs}})$. We define the overall error for the Q queries by one of the following error measures:

1. The *1-norm average error*:

$$\|e\|_1 = \frac{1}{Q}\sum_{i=1}^{Q} e_i.$$

2. The *2-norm average error*:

$$\|e\|_2 = \sqrt{\frac{1}{Q}\sum_{1\le i\le Q} e_i{}^2}.$$

3. The *infinity-norm average error*:

$$\|e\|_\infty = \max_{1\le i\le Q}\{e_i\}.$$

These error measures are special cases of the p-norm average error, for $p > 0$:

$$\|e\|_p = \Big(\frac{1}{Q}\sum_{1\le i\le Q} e_i{}^p\Big)^{1/p}.$$

The first step in thresholding is weighting the coefficients in a certain way (which corresponds to using a particular basis, such as an orthonormal basis, for example). In particular, for the Haar basis, normalization is done by dividing the wavelet coefficients $\widehat{S}(2^j), \ldots, \widehat{S}(2^{j+1}-1)$ by $\sqrt{2^j}$, for $0 \le j \le \log N - 1$. Given any particular weighting, we propose the following different thresholding methods:

1. Choose the m largest (in absolute value) wavelet coefficients.
2. Choose m wavelet coefficients in a greedy way. For example, we might choose the m largest (in absolute value) wavelet coefficients and then repeatedly do the following two steps m times:
   (a) Choose the wavelet coefficient whose inclusion leads to the largest reduction in error.
   (b) Throw away the wavelet coefficient whose deletion leads to the smallest increase in error.

   Another approach is to do the above two steps repeatedly until a cycle is reached or improvement is small.

Several other variants of the greedy method are possible:

3. Start with the $m/2$ largest (in absolute value) wavelet coefficients and choose the next $m/2$ coefficients greedily.
4. Start with the $2m$ largest (in absolute value) wavelet coefficients and throw away m of them greedily.

The straightforward method of performing each iteration of the greedy method requires $O(N^2)$ time, and thus the total time is $O(mN^2)$. By maintaining a special dynamic programming tree structure, we can speed up the preprocessing significantly.

Theorem 1 *We can greedily choose m coefficients for any of the standard error measurements in $O(N(\log N)\log m)$ time and $O(N)$ space. If Method 4 is used, it can be done in $O(N\log^2 m)$ time and $O(N)$ space.*

Proof Sketch: For simplicity, we consider the case when Method 4 is used. The other proofs are similar.

We build an "error tree" of the wavelet transform. The leaves of the tree correspond to the original signal values, and the internal nodes correspond to the wavelet coefficients. Figure 1 is the error tree for $N = 8$; each node is labeled with the wavelet coefficient or signal value that it corresponds to. The wavelet coefficient associated with an internal node in the error tree contributes to the signal values at the leaves in its subtree. For each of the $2m$ nodes that correspond to the $2m$ largest wavelet coefficients, we store the error change introduced by deleting this coefficient.

At the ith ($1 \le i \le m$) step of the greedy thresholding, we throw away the wavelet coefficient whose deletion causes the smallest increase in error. Suppose this coefficient corresponds to node n_i in the error tree. We need to update the error information of all the "relevant" nodes after the deletion. The relevant nodes fall into two classes: (a) the wavelet coefficients in the subtree rooted at n_i and (b) the wavelet coefficients on the path from n_i up to the root of the error tree.

Suppose the subtree rooted at n_i has k' leaves and m' class (a) wavelet coefficients. The maximum number of class (b) wavelet coefficients is at most $\log \frac{N}{k'}$. The important point is that the time to update a wavelet coefficient is proportional to the number of leaves in its subtree that change value. By a convexity argument, the worst-case locations for the m' class (a) wavelet coefficient are in the top $\log m'$ levels of n_i's subtree. The resulting time to update the m' class (a) wavelet coefficients is $O(k'\log m')$. The time to update the class (b) wavelet coefficients is $O(k'\log \frac{N}{k'})$.

By a convexity argument, over the m deletions, the worst case is for the m deleted wavelet coefficients to be in the top $\log m$ levels of the error tree. In this case, the m terms of $k'\log m'$ and the m terms of $k'\log\frac{N}{k'}$ sum to $O(N\log^2 m)$. □

We can use dynamic programming techniques to get further improvements for the p-norm average error for even p:

Theorem 2 *For the p-norm average error measure for any even p, we can reduce the time bound in Theorem 1 to $O(N\log m)$; for Method 4, the time is $O(N)$.*

It is well known if the wavelet basis functions are orthonormal that Method 1 is provably optimal for

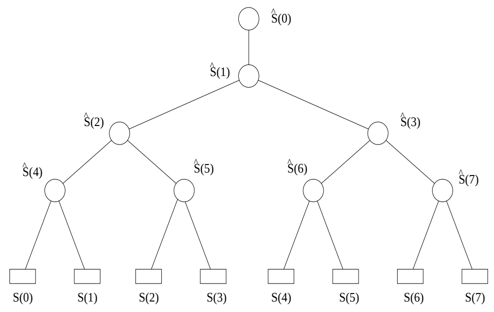


Figure 1: Error tree for $N = 8$

the 2-norm average error measure. However, for non-orthogonal wavelets like our linear wavelets and for norms other than the 2-norm, no efficient technique is known for how to choose the m best wavelet coefficients, and various approximations have been studied [2].

Our experiments show that Method 2 does best overall in terms of accuracy for wavelet-based histograms. Method 1 is easier to compute but does not perform quite as well.

4 On-Line Reconstruction

In the query phase, a range query $a \le X \le b$ is presented. We reconstruct the approximate cumulative frequencies of $a-1$ and b, denoted by c'_{a-1} and c'_b, using the m wavelet coefficients. The size of the query is estimated to be $c'_b - c'_{a-1}$. The time for reconstruction is crucial in the on-line phase:

Theorem 3 *For a given range query $a \le X \le b$, the cumulative frequencies of $a-1$ and b can be reconstructed from the m wavelet coefficients using an $O(m)$-space data structure in time $O(\log m + \#\text{ of relevant coefficients}) = O(\min\{m, \log N\})$.*

Proof Sketch: The method for reconstructing the frequency for a given domain element from the wavelet coefficients consists of identifying the $O(\log N)$ coefficients that are involved in the reconstruction. Each wavelet coefficient contributes to the reconstruction of the frequencies in a contiguous set of the domain. In a Haar wavelet, for example, each coefficient contributes a negative additive term to the reconstructed frequency for each domain value within a certain interval in the domain, and it contributes the opposite (positive) term for each value within an adjacent interval in the domain.

In particular, as demonstrated in formulas (1)–(4) of Section 3.2 for the case $N = 4$ and by the error tree of Figure 1 for the case $N = 8$, the wavelet coefficient $\widehat{S}(0)$ contributes as a positive additive term to each leaf value $S(0)$, $S(1)$, …, $S(N-1)$ in its subtree.

The wavelet coefficient $\widehat{S}(1)$ contributes as a negative term to the leaves $S(0), \ldots, S(\frac{N}{2}-1)$ in its left subtree and as a positive term to the leaves $S(\frac{N}{2}), \ldots, S(N-1)$ in its right subtree. The wavelet coefficient $\widehat{S}(2)$ contributes as a negative term to $S(0), \ldots, S(\frac{N}{4}-1)$ and as a positive term to $S(\frac{N}{4}), \ldots, S(\frac{N}{2}-1)$. The wavelet coefficient $\widehat{S}(3)$ contributes as a negative term to $S(\frac{N}{2}), \ldots, S(\frac{3N}{4}-1)$ and as a positive term to $S(\frac{3N}{4}), \ldots, S(N-1)$.

For higher-order wavelets (like linear wavelets), the contribution of a given wavelet coefficient can be represented by a constant number of adjacent intervals in the domain; unlike in the Haar case, the contribution of a given wavelet coefficient varies from point to point within each interval, but its contribution within an interval is specified by a polynomial function (which is linear in the case of linear wavelets).

The $O(m)$ intervals can be stored in linear space in an interval tree data structure [17]. Given a domain element, the wavelet coefficients corresponding to the element's frequency can be found in $O(\log m)$ time by a stabbing query on the intervals stored in the interval tree. The reconstructed frequency is then the sum of the contributions of each of those wavelet coefficients. □

Often it is useful to represent the histogram as an explicit piecewise smooth function rather than as m wavelet coefficients. For Haar wavelets the resulting function is a step function with at most $3m$ steps in the worst case, and for linear wavelets the function is piecewise linear with at most $5m$ changes in slope in the worst case. In real-life data, we can expect that the number of steps or segments is very close to m (in many cases exactly m). This property has been confirmed by an extensive set of experiments. Previous methods for expressing the histogram as a piecewise smooth function required $O(N)$ time, although some researchers suspected that $O(m \log N)$-time algorithms were possible [21]. We have developed an efficient and practical technique using priority queues that offers a substantial speedup:

Theorem 4 *The wavelet-based histogram can be transformed into a standard piecewise smooth representation in time $O(m \times \min\{\log m, \log\log N\})$ and using $O(m)$ space.*

Proof Sketch: By the reasoning behind the proof of Theorem 3, the reconstructed frequency is a polynomial function (which is constant for Haar wavelets and linear for linear wavelets) as long as the interval boundaries associated with the wavelet coefficients are not crossed. There are at most three interval boundaries per Haar wavelet coefficient and at most five interval boundaries per linear wavelet coefficient. We refer to the domain values where such boundaries occur for a given wavelet coefficient as the coefficient's *event points*.

We group the coefficients into the $\log N$ levels corresponding to the multiresolution wavelet decomposition. Let Δ be the maximum number of wavelet coefficients that can overlap another wavelet coefficient from the same level. We say that two coefficients overlap if there is a domain value whose frequency they both contribute to. For Haar wavelets we have $\Delta = 0$, and for any fixed order wavelet we have $\Delta = O(1)$. We construct the histogram by inserting into a priority queue the first event point for the first $\Delta + 1$ wavelet coefficients at each level. The polynomial function describing the reconstructed frequency does not change until the domain value corresponding to an event point is reached. We can find the next event point by performing a *delete_min* operation on the priority queue, at which point we insert into the priority queue the next event point for the coefficient involved in the *delete_min*. The polynomial function represented by the histogram is updated at each event point.

The desired time bound follows because each *delete_min* and *insert* operation takes time logarithmic in the size of the priority queue, which consists of at most $\min\{m, (\Delta + 1)\log N\}$ values at any time. □

5 Multi-Attribute Histograms

We extend the definitions in Section 1 to the multidimensional case in which there are multiple attributes. Suppose the number of dimensions is d and the attribute set is $\{X_1, X_2, \ldots, X_d\}$. Let $D_k = \{0, 1, \ldots, N_k - 1\}$ be the domain of attribute X_k. The value set V_k of attribute X_k is the set of n_k values of X_k that are present in relation R. Let $v_{k,1} < v_{k,2} < \cdots < v_{k,n_k}$ be the individual n_k values of V_k. The data distribution of X_k is the set of pairs $\mathcal{T}_k = \{(v_{k,1}, f_{k,1}), (v_{k,2}, f_{k,2}), \ldots, (v_{k,n_k} f_{k,n_k})\}$. The *joint frequency* $f(i_1, \ldots, i_d)$ of the value combination $(v_{1,i_1}, \ldots, v_{d,i_d})$ is the number of tuples in R that contain $v_{i_k,k}$ in attribute X_k, for all $1 \le k \le d$. The *joint data distribution* $\mathcal{T}_{1,\ldots,d}$ is the entire set of (*value combination, joint frequency*) pairs. The joint frequency matrix $\mathcal{F}_{1,\ldots,d}$ is a $n_1 \times \cdots \times n_d$ matrix whose $[i_1, \ldots, i_d]$ entry is $f(i_1, \ldots, i_d)$. We can define the *cumulative joint distribution* $\mathcal{T}^{\mathcal{C}}_{1,\ldots,d}$ and *extended cumulative joint distribution* $\mathcal{T}^{\mathcal{C}^+}_{1,\ldots,d}$ by analogy with the one-dimensional case. The extended cumulative joint frequency $\mathcal{F}^{\mathcal{C}^+}_{1,\ldots,d}$ for the d attributes $X_1, X_2, \ldots, X_d$ is a $N_1 \times N_2 \times \cdots \times N_d$ matrix p defined by

$$p[x_1, x_2, \ldots, x_d] = \sum_{i_1=0}^{x_1} \sum_{i_2=0}^{x_2} \cdots \sum_{i_d=0}^{x_d} f(i_1, i_2, \ldots, i_d).$$

When a query involves multiple attributes in a relation, the selectivity depends on these attributes' *joint data distribution*, that is, the frequencies of all combination of attribute values. To simplify the estimation of the query result size, most commercial DBMSs make the *attribute value independent assumption*. Under such an assumption, a system maintains histograms only on individual attributes, and the joint probabilities are derived by multiplying the individual probabilities. Real-life data rarely satisfy the attribute value independent assumption. Functional dependencies and various types of correlations among attributes are very common. Making the attribute value independent assumption in these cases results in very inaccurate estimation of the joint data distribution and poor selectivity estimation.

5.1 Previous Approaches

Muralikrishna and DeWitt [11] use an interesting spatial index partitioning technique for constructing equidepth histograms for multidimensional data. One drawback with this approach is that it considers each dimension only once during the partition. Poosala and Ioannidis [15] propose two effective alternatives. The first approach partitions the joint data distribution into mutually disjointed buckets and approximates the frequency and the value sets in each bucket in a uniform manner. Among this new class of histograms, the multidimensional MaxDiff(V,A) histograms computed using the MHIST-2 algorithm are most accurate and perform better in practice than previous methods [15]. The second approach uses the powerful *singular value decomposition* (SVD) technique from linear algebra, which is limited to handling two dimensions. Its accuracy depends largely on that of the underlying one-dimensional histograms.

5.2 Using Multidimensional Wavelets

A very nice feature of our wavelet-based histograms is that they extend naturally to multiple attributes by means of multidimensional wavelet decomposition and reconstruction. The procedure of building the multidimensional wavelet-based histogram is similar to that of the one-dimensional case except that we approximate the *extended cumulative joint distribution* $\mathcal{T}^{\mathcal{C}^+}_{1,\ldots,d}$ instead of $\mathcal{T}^{\mathcal{C}^+}$.

In the preprocessing step, we obtain the joint frequency matrix $\mathcal{F}_{1,\ldots,d}$ and use it to compute the extended cumulative joint frequency matrix $\mathcal{F}^{\mathcal{C}^+}_{1,\ldots,d}$. We then use the multidimensional wavelet transform to decompose $\mathcal{F}^{\mathcal{C}^+}_{1,\ldots,d}$. Finally, thresholding is performed to obtain the wavelet-based histogram.

In the query phase, in order to approximate the selectivity of a range query of the form $(a_1 \le X_1 \le b_1) \wedge \cdots \wedge (a_d \le X_d \le b_d)$, we use the wavelet coefficients to reconstruct the 2^d cumulated counts $p[x_1, x_2, \ldots, x_d]$, for $x_j \in \{a_j - 1, b_j\}$, $1 \le j \le d$. The following theorem adopted from [7] can be used to compute an estimate S' for the result size of the range query:

Theorem 5 ([7]) *For each $1 \le j \le d$, let*

$$s(j) = \begin{cases} 1 & \text{if } x_j = b_j; \\ -1 & \text{if } x_j = a_j - 1. \end{cases}$$

Then the approximate selectivity for the d-dimensional range query specified above is

$$S' = \sum_{\substack{x_j \in \{a_j - 1, b_j\} \\ 1 \le j \le d}} \prod_{i=1}^{d} s(i) \times p[x_1, x_2, \ldots, x_d].$$

By convention, we define $p[x_1, x_2, \ldots, x_d] = 0$ if $x_j = -1$ for any $1 \le j \le d$.

6 Empirical Results

In this section we report on some experiments that compare the performance of our wavelet-based technique with those of Poosala et al [16, 13, 15] and random sampling. Our synthetic data sets are those from previous studies on histogram formation and from the TPC-D benchmark [22]. They correspond to studies on typical data found on the web. For simplicity and ease of replication, we use method 1 for thresholding in all our wavelet experiments.

6.1 Experimental Comparison of One-Dimensional Methods

In this section, we compare the effectiveness of wavelet-based histograms with MaxDiff(V,A) histograms and random sampling. Poosala et al [16] characterized the types of histograms in previous studies and proposed new types of histograms. They concluded in their experiments that the MaxDiff(V,A) histograms perform best overall.

Random sampling can be used for selectivity estimation [5, 6, 10, 9]. The simplest way of using random sampling to estimate selectivity is, during the off-line phase, to take a random sample of a certain size (depending on the catalog size limitation) from the relation. When a query is presented in the on-line phase, the query is evaluated against the sample, and the selectivity is estimated in the obvious way: If the result size of the query using a sample of size t is s, the selectivity is estimated as sT/t, where T is the size of the relation.

Our one-dimensional experiments use the many synthetic data distributions described in detail in [16]. We use $T = 100{,}000$ to $500{,}000$ tuples, and the number n of distinct values of the attribute is between 200 and 500. The distributions from [16] subsume the types of one-dimensional distributions from the TPC-D benchmark.

We use eight different query sets in our experiments:

A: $\{X \le b \mid b \in D\}$.
B: $\{X \le b \mid b \in V\}$.
C: $\{a \le X \le b \mid a, b \in D,\ a < b\}$.
D: $\{a \le X \le b \mid a, b \in V,\ a < b\}$.
E: $\{a \le X \le b \mid a \in D,\ b = a + \Delta\}$, where Δ is a positive integer constant.
F: $\{a \le X \le b \mid a \in V,\ b = a + \Delta\}$, where Δ is a positive integer constant.
G: $\{X = b \mid b \in D\}$.
H: $\{X = b \mid b \in V\}$.

Different methods need to store different types of information. For random sampling, we only need to store one number per sample value. The MaxDiff(V,A) histogram stores three numbers per bucket: the number of distinct attribute values in the bucket, the largest attribute value in the bucket, and the average frequency of the elements in the bucket. Our wavelet-based histograms store two numbers per coefficient: the index of the wavelet coefficient and the value of the coefficient.

In our experiments, all methods are allowed the same amount of storage. The default storage space we use in the experiments is 42 four-byte numbers (to be in line with Poosala et al's experiments [16], which we replicate); the limited storage space corresponds to the practice in database management systems to devote only a very small amount of auxiliary space to each relation for selectivity estimation [18]. The 42 numbers correspond to using 14 buckets for the MaxDiff(V,A) histogram, keeping $m = 21$ wavelet coefficients for wavelet-based histograms, and maintaining a random sample of size 42.

The relative effectiveness of the various methods is fairly constant over a wide variety of value set and frequency set distributions. We present the results from one experiment that illustrates the typical behavior of the methods. In this experiment, the spreads of the value set follow the *cusp_max* distribution with Zipf parameter $z = 1.0$, the frequency set follows a Zipf distribution with parameter $z = 0.5$, and frequencies are randomly assigned to the elements of the value set.[1] The value set size is $n = 500$, the domain size is $N = 4096$, and the relation size is $T = 10^5$. Tables 1–5 give the errors of the methods for query sets A, C, E, G, and H. Figure 2 shows how well the methods approximate the cumulative distribution of the underlying data.

Wavelet-based histograms using linear bases perform the best over almost all query sets, data distributions, and error measures. The random sampling method does the worst in most cases. Wavelet-based histograms using Haar bases produce larger errors than MaxDiff(V,A) histograms in some cases and smaller errors in other cases. The reason for Haar's lesser performance arises from the limitation of the step function approximation. For example, in the case that both frequency set and value set are uniformly distributed, the cumulative frequency is a linear function of the attribute value; the Haar wavelet histogram produces a sawtooth approximation, as shown in Figure 2b. The Haar estimation can be improved by linearly interpolating across each step of the step function so that the reconstructed frequency is piecewise linear, but doing that type of interpolation after the fact amounts to a histogram similar to the one produced by linear wavelets (see Figure 2a), but without the explicit error optimization done for linear wavelets when choosing the m coefficients.

We also studied the effect of storage space for different methods. Figure 3 plots the result of one set of our experiments for queries from query set A. In these experiments, the value set follows *cusp_max* distribution with parameter $z = 1.0$, the frequency set follows a Zipf distribution with parameter $z = 1.0$, and frequencies are assigned to value set in a random way. The value set size is $n = 500$, the domain size is $N = 4096$, and the relation size is $T = 10^5$.

In addition to the above experiments we also tried a modified MaxDiff(V,A) method so that only two numbers are kept for each bucket instead of three (in particular, not storing the number of distinct values in each bucket), thus allowing 21 buckets per histogram instead of only 14. The accuracy of the estimation was improved. The advantage of the added buckets was somewhat counteracted by less accurate modeling within each bucket. The qualitative results, however, remain the same: The wavelet-based methods are significantly more accurate. Further improvements in the wavelet techniques are certainly possible by quantization and entropy encoding, but they are beyond the scope of this paper.

6.2 Experimental Comparison of Multidimensional Methods

In this section, we evaluate the performance of histograms on two-dimensional (two-attribute) data. We compare our wavelet-based histograms with the MaxD-

[1] The *cusp_max* and *cusp_min* distributions are two-sided Zipf distributions. Zipf distributions are described in more detail in [13]. Zipf parameter $z = 0$ corresponds to a perfectly uniform distribution, and as z increases, the distribution becomes exponentially skewed, with a very large number of small values and a very small number of large values. The distribution for $z = 2$ is already very highly skewed.

Error Norm	*Linear Wavelets*	*Haar Wavelets*	*MaxDiff(V,A)*	*Random Sampling*
$\|\|e^{\mathrm{rel}}\|\|_1$	0.6%	4.5%	8%	20%
$\|\|e^{\mathrm{abs}}\|\|_1/T$	0.16%	0.8%	3%	8%
$\|\|e^{\mathrm{abs}}\|\|_2/T$	0.26%	0.64%	3.2%	10%
$\|\|e^{\mathrm{abs}}\|\|_\infty/T$	1.5%	5.6%	11%	13%
$\|\|e^{\mathrm{comb}}\|\|_1,\ \alpha=1,\beta=100$	0.6	4.4	8	20
$\|\|e^{\mathrm{comb}}\|\|_1,\ \alpha=1,\beta=1000$	5	30	80	200
$\|\|e^{\mathrm{comb}}\|\|_2,\ \alpha=1,\beta=100$	5.1	70.4	12.8	19
$\|\|e^{\mathrm{comb}}\|\|_2,\ \alpha=1,\beta=1000$	19	224	192	243

Table 1: Errors of various methods for query set A.

Error Norm	*Linear Wavelets*	*Haar Wavelets*	*MaxDiff(V,A)*	*Random Sampling*
$\|\|e^{\mathrm{abs}}\|\|_1/T$	0.2%	1.1%	5%	3.5%
$\|\|e^{\mathrm{abs}}\|\|_2/T$	0.035%	0.18%	0.71%	0.6%
$\|\|e^{\mathrm{abs}}\|\|_\infty/T$	2.4%	10%	20%	16%

Table 2: Errors of various methods for query set C.

Error Norm	*Linear Wavelets*	*Haar Wavelets*	*MaxDiff(V,A)*	*Random Sampling*
$\|\|e^{\mathrm{abs}}\|\|_1/T$	0.1%	0.42%	0.15%	0.35%
$\|\|e^{\mathrm{abs}}\|\|_2/T$	0.19%	0.96%	0.26%	0.64%
$\|\|e^{\mathrm{abs}}\|\|_\infty/T$	1.5%	6%	3%	4.6%

Table 3: Errors of various methods for query set E with $\Delta = 10$.

Error Norm	*Linear Wavelets*	*Haar Wavelets*	*MaxDiff(V,A)*	*Random Sampling*
$\|\|e^{\mathrm{abs}}\|\|_1/T$	0.03%	0.04%	0.04%	0.04%
$\|\|e^{\mathrm{abs}}\|\|_2/T$	0.077%	0.32%	0.096%	0.24%
$\|\|e^{\mathrm{abs}}\|\|_\infty/T$	1.6%	7%	2%	4.6%

Table 4: Errors of various methods for query set G.

Error Norm	*Linear Wavelets*	*Haar Wavelets*	*MaxDiff(V,A)*	*Random Sampling*
$\|\|e^{\mathrm{abs}}\|\|_1/T$	0.03%	0.42%	0.2%	0.4%
$\|\|e^{\mathrm{abs}}\|\|_2/T$	7.7%	16.1%	25.6%	38%
$\|\|e^{\mathrm{abs}}\|\|_\infty/T$	0.2%	7%	2%	5 %

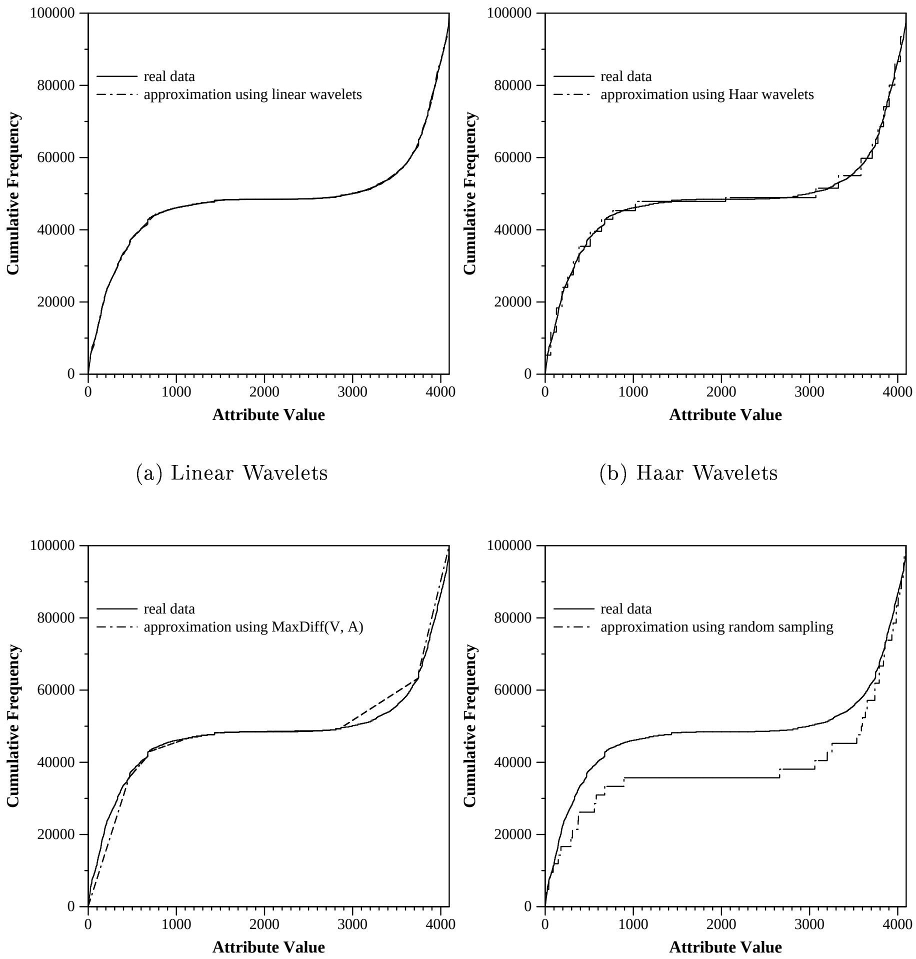
Table 5: Errors of various methods for query set H.

Data Range	*Linear Wavelets*	*Haar Wavelets*	*MHIST-2*
255	0.3%	1.5%	7%
511	0.3%	1.6%	8%
1023	0.3%	1.6%	6%
2047	0.3%	1.6%	6%

Table 6: $\|\|e^{\mathrm{abs}}\|\|_1/T$ errors of various two-dimensional histograms for TPC-D data.

iff(V, A) histograms computed using the MHIST-2 algorithm [15] (which we refer to as MHIST-2 histograms).

In our experiments we use the synthetic data described in [15], which is indicative of various real-life



(a) Linear Wavelets

(b) Haar Wavelets

(c) MaxDiff(V, A)

(d) Random Sampling

Figure 2: Approximation of the cumulative data distribution using various methods.

data [1], and the TPC-D benchmark data [22]. Our query sets are obtained by extending the query sets A–H defined in Section 6.1 to the multidimensional cases.

The main concern of the multidimensional methods is the effectiveness of the histograms in capturing data dependencies. In the synthetic data we used, the degree of the data dependency is controlled by the z value used in generating the Zipf distributed frequency set. A higher z value corresponds to fewer very high frequencies, implying stronger dependencies between the attributes. One question raised here is what is the reasonable range for that z value. As in [15], we fix the relation size T to be 10^6 in our experiments. If we assume our joint value set size is $n_1 \times n_2$, then in order to get frequencies that are at least 1, the z value cannot be greater than a certain value. For example, for $n_1 = n_2 = 50$, the upper bound on z is about 1.67. Any larger z value will yield frequency values smaller than 1. In our experiments, we choose various z in the range $0 \le z \le 1.5$. The value $z = 1.5$ already corresponds to a highly skewed frequency set; its top three frequencies are 388747, 137443, and 74814, and the 2500th frequency is 3. In [15], larger z values are considered; most of the Zipf frequencies are actually very close to 0, so they are instead boosted up to 1, with the large frequencies correspondingly lowered, thus yielding semi-Zipf distributed frequency sets [14]. The relative effectiveness of different histograms is fairly constant over a wide variety of data distributions and query sets that we studied. Figure 4 depicts the effect of the Zipf skew parameter z on the accuracy of different types of histograms for one typical set of experiments. In these experiments, we use $N_1 = N_2 = 256$ and $n_1 = n_2 = 50$; the value set in each dimension follows *cusp_max* distribution with $z_s = 1.0$. The storage space is 210 four-bytes numbers (again, to be in line with the default storage space

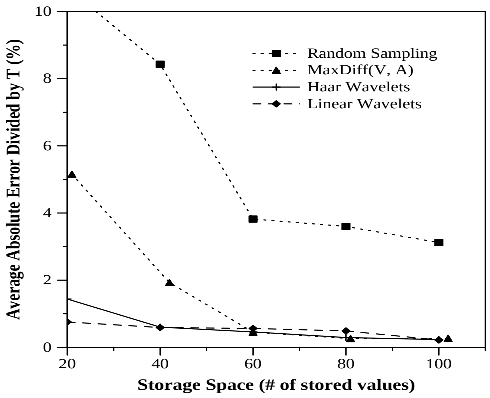


Figure 3: Effect of storage space for various one-dimensional histograms using query set A.

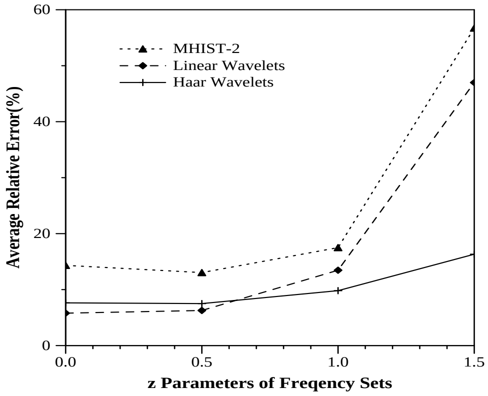
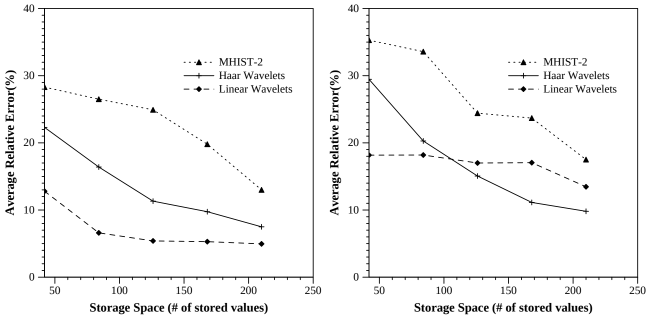


Figure 4: Effect of frequency skew, as reflected by the Zipf z parameter for the frequency set distribution.



(a) Zipf parameter z=0.5 (b) Zipf parameter z=1.0

Figure 5: Effect of storage space on two-dimensional histograms.

in [15]). It corresponds to using 30 buckets for MHIST-2 histogram (seven numbers per bucket) and keeping 70 wavelet coefficients for wavelet-based based histogram (three numbers per coefficient). The queries used are those from query set A.

In other experiments we study the effect of the amount of allocated storage space upon the accuracy of various histograms. As we mentioned above, the amount of storage devoted to a catalog is quite limited in any practical DBMS. Even without strict restrictions on the catalog size, a big catalog means that more buckets or coefficients need to be accessed in the on-line phase, which slows down performance. Figure 5 plots the effectiveness of the allocated storage space on the performance of various histograms. In the experiments, we use the same value set and query set as for Figure 4. The frequency skew is $z = 0.5$ for (a) and $z = 1.0$ for (b).

We conducted experiments using TPC-D data [22]. We report the results for a typical experiment here. In this experiment, we use the *L_SHIPDATA* and *L_RECEIPDATA* columns in the *LINEITEM* table, defined as follows:

$$L\_SHIPDATA = O\_ORDERDATA + random(121)$$
$$L\_RECEIPTDATA = L\_SHIPDATA + random(30),$$

where $O\_ORDERDATA$ is uniformly distributed between $STARTDATA$ and $ENDDATA - 151$ and $random(n)$ returns a random value between 1 and n. We fix the table size to be $T = 10^6$ and vary the size n of the value set V by means of changing *data range*, the difference between $ENDDATA$ and $STARTDATA$. Table 6 shows the $\|e^{\mathrm{abs}}\|_1/T$ errors of the different histogram methods for Set A queries.

7 Conclusions

In this paper we have proposed a method to build efficient and effective histograms using wavelet decomposition. Our histograms give improved performance for selectivity estimation compared with random sampling and previous approaches.

In [26], a new thresholding method based on a logarithm transform is proposed that dramatically reduces the errors in wavelet-based approximation of high-dimensional data, such as in OLAP data cubes. Experiments show that by applying the new thresholding method in building wavelet-based histograms, we can achieve much better accuracy even for the low-dimensional data considered in Section 6; the relative errors reported in Section 6 can be cut dramatically by a factor of 3 in typical cases, and the absolute errors are usually reduced by more than half.

High-dimensional data can often be much larger than the low-dimensional data considered in this paper, and I/O communication can be a bottleneck. I/O efficient techniques for computing the wavelet transform and thresholding for high-dimensional data are discussed in [26].

Ongoing work deals with improved space-accuracy tradeoffs by quantizing coefficients using, for example, a generalized zero-tree [19] followed by entropy encoding of the quantized coefficients. It may be that other wavelet bases perform better in practice than do the Haar and linear wavelet bases we have considered in this paper, and those possibilities will be considered on real-life data.

Wavelet-based histograms should serve as an effective synopsis data structure for selectivity estimation in the context of the on-line summary mechanism of [3]. We are developing efficient algorithms for maintaining the wavelet-based histograms given insertions and deletions in the underlying relation.

Acknowledgments. We gratefully acknowledge earlier discussions with Christos Faloutsos, Suleyman Cenk Sahinalp, Wim Sweldens, and Brani Vidacovic. We especially thank Vishy Poosala for much useful background and information on histogram techniques.

References

[1] Census Bureau Databases. http://www.census.gov/.

[2] D. L. Donoho. Unconditional bases are optimal bases for data compression and statistical estimation. Technical report, Department of Statistics, Stanford University, 1992.

[3] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998.

[4] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proceedings of the 23rd VLDB Conference*, Athens, Greece, August 1997.

[5] P. Haas and A. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the 1992 ACM SIGMOD Conference*, 1992.

[6] P. Haas and A. Swami. Sampling-based selectivity for joins using augmented frequent value statistics. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, March 1995.

[7] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range queries in OLAP data cubes. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, May 1997.

[8] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36(3):377–412, 1994.

[9] R. Lipton and J. Naughton. Query size estimation by adaptive sampling. *J. of Comput. Sys. Sci.*, 51:18–25, 1985.

[10] R. Lipton, J. Naughton, and D. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceeding of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 1–11, 1990.

[11] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 28–36, 1988.

[12] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 256–276, 1984.

[13] V. Poosala. *Histogram-Based Estimation Techniques in Database Systems.* Ph. D. dissertation, University of Wisconsin-Madison, 1997.

[14] V. Poosala. Personal communication, 1997.

[15] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the 23rd VLDB Conference*, Athens, Greece, August 1997.

[16] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, May 1996.

[17] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[18] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.

[19] J. M. Shapiro. An embedded wavelet hierarchical image coder. In *Proceedings of 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 657–660, San Francisco, CA, March 1992.

[20] E. J. Stollnitz, T. D. Derose, and D. H. Salesin. *Wavelets for Computer Graphics.* Morgan Kaufmann, 1996.

[21] W. Sweldens. Personal communication, 1997.

[22] TPC benchmark D (decision support), 1995.

[23] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, March 1985.

[24] J. S. Vitter. An efficient algorithm for sequential random sampling. *ACM Transactions on Mathematical Software*, 13(1):58–67, March 1987.

[25] J. S. Vitter. External memory algorithms. In *Proceedings of the 1998 ACM Symposium on Principles of Database Systems*, June 1998. Invited tutorial.

[26] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation via wavelets. Manuscript, 1998.