TRCS00-15

The Golden Estimator: Efficient Range Query Estimation

Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi

Department of Computer Science
University of California, Santa Barbara, CA 93106
Email:{ywu, agrawal, amr}@cs.ucsb.edu

Abstract. Query size estimation is crucial for many database system components. In particular, query optimizers need efficient and accurate query size estimation when deciding among alternative query plans. In this paper we propose the *Golden Estimator*, which is based on the so called golden rule of sampling proposed by von Neumann, for estimating the size of single dimensional range queries. The *Golden Estimator* randomly samples the frequency domain using the cumulative frequency distribution. We argue why this approach will yield good estimates irrespective of the actual underlying distribution of values. We then experimentally show that the *Golden Estimator* gives better approximation than state of the art histogram based and wavelet based approaches under the same space requirement.

Key words. random sampling, cumulative frequency distribution, query estimation, range query

1 Introduction

Query size estimation plays an important role in the query optimizer of a database management system. Depending on the size of the intermediate results, a particularly query execution plan may be favored over another. A common approach in relational databases is to store, for each attribute, summary information that can be used to estimate the resulting size of a query. In this paper, we are particularly interested in estimating the size of selection or range queries that are defined over a single attribute of a relational table.

Random samples of tuples from the base relation of database can be used for selectivity estimation [HS92, HS95, GM98, HHW97, HH99, LNS90]. The AQUA system [GPA+98, GM98, AGPR99] uses random samples of tuples for general purpose query result estimation. The idea to to create a down-sized copy of the original relation and run the queries against the down-sized copy, which is significantly smaller than the original relation, to get an estimate of the results. Recently Chaudhuri et al. [CMN99] have developed a theoretical framework for explaining the difficulties in implementing sampling as a primitive relational operator. We refer to the above techniques as *tuple sampling*.

Instead of using a down-sized copy of the original relation for query estimation, other stored information (e.g., histograms) that summarizes the distribution of attribute values of a relation can be used for query result estimation. Such approaches further reduce the amount of auxiliary information maintained for estimation. Piatetsky-Shapiro et al. [PSC84] proposed the histogram method known today as the equal depth histogram. Poosala et al. [PIHS96, Poo97] have developed a general taxonomy for classifying histogram methods. In a histogram, the attribute values are divided into buckets. Jagadish et al. [JKM+98] used a dynamic programming approach to compute optimal histogram. An alternative approach using multiresolutional wavelet decomposition to build histograms was proposed [VWI98, MVW98, VW99]. Empirical results indicated that the wavelet based approach provides better estimation for range queries.

One of the main problems of applying estimation techniques in a database management system is the unknown characteristic of the distribution of the data that will populate a given DBMS. In fact, in order for such a technique to be useful, the estimation technique should be effective for different kinds of data distributions. This is because a DBMS is used for a variety of applications resulting in a wide spectrum of data that populates the database. In this paper, we develop a new strategy to summarize the distribution of attribute values, the *Golden Estimator*, based on a simple observation (known as the golden rule of sampling) made by von Neumann [Eck87] for sampling cumulative probability distributions. The proposed method uses samples based on the frequency domain of the cumulative frequency distribution. The surprising observation is that such an approach results in samples that reflect the distribution without any a priori knowledge. We develop this approach in the discrete domain and conduct various experiments to compare it with other state of the art approaches. We demonstrate that in most cases, under the same space requirement, the *Golden Estimator* gives more accurate estimation for the result size of range queries.

This paper is organized as follows. In Section 2 we introduce our terminology and discuss in detail how to apply a simple statistical rule for random sampling to estimate range queries in the discrete domain. Extensive experiments and results are presented in Section 3. Section 4 concludes this paper.

2 Summarization Using Random Sampling

In this section, we present a summarization approach that uses random sampling over the distribution function of a relation instead of the tuple level as in the tuple sample techniques. We begin by defining the range query estimation problem and then describe a random sampling technique that is invariant to the underlying data distribution. Unfortunately, this technique is applicable only in the continuous domain. We therefore adapt this approach for discrete domains.

2.1 Preliminaries

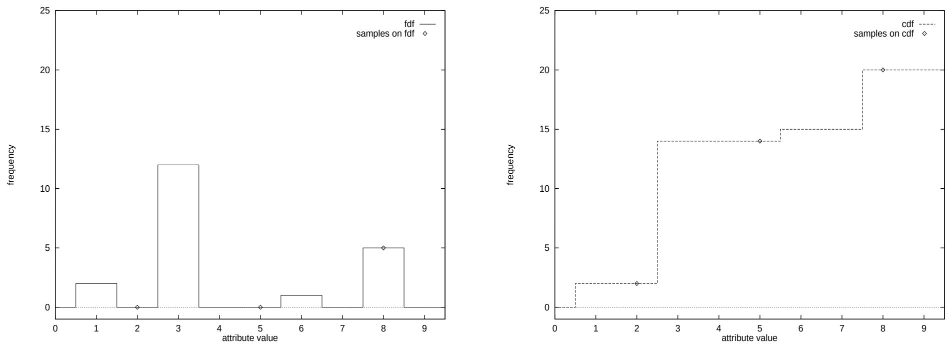
Given a relation R with an attribute A, we are interested in estimating the size of a range query on attribute A. We adopt the notation develop in [PIHS96, MVW98] where information about the attribute domain is summarized and used in estimating the size of the query. We assume the discrete domain D of attribute A is $\{d_0, d_1, d_2, ..., d_{N-1}\}$ which contains all the possible values of A. A range query is specified using a range predicate of the form $a \leq A \leq b$, where a and b are constant values in the discrete domain D. To calculate the size of this query, without actually executing it, a frequency distribution function is needed. The *frequency* f_i of value d_i for attribute A is the number of tuples $t \in R$ with $t.A = d_i$. If no tuples t in R have $t.A = d_i$, then $f_i = 0$; i.e., zero frequency. The *frequency distribution function* (fdf) is an ordered set of value pairs $fdf = \{\langle d_0, f_0\rangle, \langle d_1, f_1\rangle, \langle d_2, f_2\rangle, ..., \langle d_{N-1}, f_{N-1}\rangle\}$. Given the fdf, the size of a range query $q(a \leq A \leq b)$ can be calculated as the number of tuples $t \in R$ with $a \leq t.A \leq b$, which is equal to $\sum_{i=a}^{b} f_i$. Much research has been devoted to obtaining accurate values for the fdf [HS95, Vit85, AD85, PIHS96], however good approximations usually require high storage, which, is unrealistic to use the fdf to estimate query sizes. As a result, most commercial systems maintain histograms for estimating query sizes and recently there have been several histogram based techniques that have been proposed [PIHS96, PI97, Poo97]. In these approaches, the histogram is derived either from the actual fdf or, more realistically, based on a down-sized copy (often obtained using tuple sampling) with lesser accuracy.

In this paper, we will use sample pairs $\langle d_i, f_i\rangle$ to approximate the fdf (as with histogram derivations, we can either use the actual fdf or a down-sized copy). Our approach is based on the cumulative frequency distribution, which can be easily obtained from the frequency distribution function. The main challenge in query size estimation is to use a limited amount of storage to approximate the fdf. This approximation is then used to estimate the number of tuples that are in the range specified by the query.

2.2 Random Sampling in Databases

Random sampling is one of the most straightforward and simple techniques used for estimating values in a given distribution. In general, random sampling techniques work as follows. In the construction phase, random samples are selected from the domain under study, in our case, for example, the frequency distribution (fdf), and stored as value pairs $\langle d_i, f_i\rangle$ in a searchable structure (e.g., array, tree, etc.). Note that unlike tuple sampling, where actual tuples are sampled, in this approach the domain is sampled and hence some values may be zero. We are trying to estimate the entire distribution. In the query phase, when a range query $q(a \leq X \leq b)$ is presented we search the stored samples and compute the estimated result s' from the samples corresponding to the range $[a : b]$. The absolute difference between the actual result size s and s', $|s - s'|$ is referred to as the *absolute error*. The *relative error* E_{rel}, which evaluates the error as related to the result size is defined as:

$$E_{rel} = \frac{|s - s'|}{s} \times 100 \tag{1}$$



(a) fdf (b) cdf

Fig. 1. Example fdf and cdf

Figure 1.(a) shows an example fdf. One naive approach is to construct the fdf by randomly sampling the attribute domain and when a query is issued, the sampled fdf value pairs are used to estimate the result size. For example, let the sample set taken be $S_{fdf} = \{\langle 2, 0\rangle, \langle 5, 0\rangle, \langle 8, 5\rangle\}$, which is randomly sampled from the fdf by attribute domain as shown in Figure 1.(a). Consider a range query $q(2 \le X \le 7)$. Since two samples in this query fall in the query range, namely $\langle 2, 0\rangle$ and $\langle 5, 0\rangle$, we can therefore use these two samples to estimate the query result. If we assume a uniform value distribution (since we do not assume knowledge of the underlying distribution), we can assume that neighboring attributes have similar frequencies. We therefore estimate the frequencies of attribute values 3, 4, 6, and 7 all are zero because their neighboring attributes 2 and 5 have zero frequencies, which yields a query result estimate of zero. As can be verified from Figure 1.(a), the actual result to this query is 13. Hence the relative error $E_{rel} = |13 - 0|/13 \times 100 = 100\%$. In fact, several disadvantages can be observed when using randomly sampled sets from the fdf. First, as shown in Figure 1.(a), most of the frequencies are of a very small value, zeros or ones in most cases. The sample set is expected to have most of the value pairs with zero frequencies and thus will not contribute to the estimate. Second, to use the value pairs in the sample set to estimate the missing neighboring values is inappropriate because the uniformity distribution assumption does not hold in general; i.e., there may be no correlation among frequencies of neighboring attributes.

As was observed by Matias et al. [MVW98], a more appropriate approach is to randomly sample the cumulative distribution function (cdf) instead of the fdf. The cdf is the ordered set of value pairs $\{\langle d_0, c_0\rangle, \langle d_1, c_1\rangle, \langle d_2, c_2\rangle, ..., \langle d_{N-1}, c_{N-1}\rangle\}$ where c_i is the cumulative frequency of value d_i, which is the number of tuples $t \in R$ with $t.A \le d_i$; i.e., $c_i = \sum_{j=1}^{i} f_j$. For a range query $q(a \le A \le b)$, the cdf can be used to compute the number of tuples as $c_b - c_a$. The intuition behind this is that each sample of the cdf $\langle d_i, c_i\rangle$ contains the sum of all frequencies of the fdf that is smaller than d_i; i.e., even if the random sample misses some attribute values with large frequencies, the missing information can still be gathered in a random sample with larger d_i. Furthermore, since the cdf is a monotonically increasing function, it is easier to have better approximations than using the fdf. From a query cost point of view, it is worth noting that the worst case complexity of executing a range query using the cdf is $O(1)$ whereas using the fdf is $O(n)$. This property was exploited in the context of multidimensional data cubes by Ho et al. [HAMS97] to ensure constant query cost.

To show how to apply randomly sampled cdf for range query estimations, Figure 1.(b) shows the cdf for the underlying fdf. Let us take the same three samples as the above but from the cdf, $S_{cdf} =$

$\{\langle 2,2\rangle, \langle 5,14\rangle, \langle 8,20\rangle\}$. The estimated query result of $q(2 \leq X \leq 7)$ is $c'_7 - c'_2$, where c'_2 and c'_7 are estimates of the actual values c_2 and c_7, respectively. Because the cdf with attribute value 2 is contained in the sample set, c'_2 can be determined exactly as $c'_2 = c_2 = 2$. c'_7 can be estimated in many ways. The simplest way to estimate the value of c'_7 is to assign it to the nearest sample point in the sample set S_{cdf}, which is $c_8 = 20$. The final estimation is now $c'_7 - c'_2 = 20 - 2 = 18$. The relative error is $E_{rel} = |13-18|/13 \times 100 = 38.5\%$, which is smaller then the previous estimate using the random sample set of fdf. But choosing the nearest neighbor point $\langle d_i, c_i\rangle$ in the sample set to represent the query point c'_j in some cases is not efficient. In this example, to estimate c'_7 we used the nearest neighbor $c_8 = 20$ in the sample set S_{cdf} but the actual c_7 is 15 and thus the error is 5.

One can reduce the error of the estimation further by using a more elaborate method, namely, *interpolation*. For example, to estimate c'_i, we take the two nearest neighbor samples $\langle d_{low}, c_{low}\rangle$ and $\langle d_{high}, c_{high}\rangle$ from the sample set S_{cdf} such that d_{low} is the largest value smaller than or equal to d_i and d_{high} is the smallest value greater than or equal to d_i. To interpolate c_i, we use

$$c'_i = c_{low} + (c_{high} - c_{low}) \times \frac{d_i - d_{low}}{d_{high} - d_{low}} \tag{2}$$

In the above example, to estimate c'_7 we take two samples from S_{cdf} $\langle d_{low}, c_{low}\rangle = \langle 5,14\rangle$ and $\langle d_{high}, c_{high}\rangle = \langle 8,20\rangle$. Using Equation 2, $c'_7 = 14 + (20-14) \times (7-5)/(8-5) = 18$. As a result, by using the interpolation method together with the random sampling of the cdf, the E_{rel} reduces to $|13-16|/13 \times 100 = 23\%$.

Random sampling of the cdf gives better estimation than sampling the fdf in the above example because sampling the fdf has the potential of choosing sample sets that contain less information if the distribution of the fdf value pairs is sparse with respect to the attribute domain. Sampling the cdf instead of fdf can partially remedy this problem. Although sampling the cdf instead of fdf yields better estimation as argued earlier, several problems still remain with regard to estimation based on random sampling. For example, with the above cdf sample set, query $q(0 \leq X \leq 3)$ will result in an estimated answer of zero because $\langle d_2, c_2\rangle$ is the nearest sample to both d_0 and d_3. The error can be greatly reduced if the sample set contains the sample $\langle 3,14\rangle$. The above example illustrates one of the main problem with random sampling, namely, that sampling is performed in a random manner whereas the distribution of the underlying data may not be uniform. Hence the sample set may not capture the necessary information in the data. Also note that sampling on the cdf does not always give better estimates than sampling on fdf. This is especially true when the underlying distribution has near uniform or clusters of uniform distributions. What is needed is a method that samples the data in a manner that matches the underlying distribution.

One approach to achieve this is to sample more values with higher frequencies than values with lower frequencies; i.e., samples should not be taken randomly by attribute values but according to their corresponding frequencies. However, in general, such a sampling strategy is hard if the underlying distribution is not known in advance. Alternatively, the fdf values can be sorted by frequencies and then apply a biased sampling;i.e., more samples are taken from the more frequent values. In fact, Poosala et al. [IP95, PIHS96] have shown that in constructing histograms, the best results are attained when values are *sorted* and histogram buckets are chosen to favor more frequent values. In the following section we describe a simple approach that captures the essence of the underlying distribution without a priori knowledge of the distribution and without the need for sorting.

2.3 The Golden Rule for Random Sampling

We now propose to adapt a very efficient sampling mechanism for capturing the relevant information, irrespective of the underlying distributions. In a letter to Ulam in 1947, von Neumann suggested a sampling rule often referred to as "the golden rule of sampling" (Los Alamos Science, p.135, June 1987, [Eck87]). Since the rule is stated for continuous probability distributions, we now digress, and assume that the distribution function is a continuous probability function, and later show how to apply the proposed technique to a discrete frequency function. The initial development is based on [Com95].

Consider a non-uniform probability distribution function $f(x)$. As discussed earlier, uniformly sampling such a distribution will not yield a set of random samples that effectively capture the essential information of $f(x)$. For example, if $f(x)$ has a Zipf distribution, sampling may yield most samples that do not carry any information or have zero values. Alternatively, assume that we have a transformation h such that $h(f(x))$ is uniform and $f(x)$ and $h(f(x))$ are 1-to-1 (i.e., $h(f(x))$ is unique and the inverse is well defined). Since $h(f(x))$ is uniformly distributed, any randomly sampled set S effectively captures its essential characteristics. Now, if we invert the set S, i.e., calculate a set of samples $h^{-1}(S)$ in the original domain $f(x)$, we have a sample set that captures the underlying probability distribution of $f(x)$.

The main challenge now is to find a transformation h that results in a uniform distribution and is a 1-to-1 function with $f(x)$. The beauty of von Neumann's observation is that the cumulative distribution function is exactly such a transformation; i.e., given a probability distribution function $f(x)$, the cumulative distribution function $F(x)$ is a uniform distribution [Eck87]. (Detailed mathematics can be found in Appendix A.) The *golden rule* of sampling can be summarized as follows.

1. Given a pdf $f(x)$, generate the cdf $F(x)$.
2. Generate a random number y between $[0, c_{N-1}]$ (the range of the cdf is between 0 and c_{N-1}) then equate it to the cdf; i.e., $y = F(x)$.
3. Solve for x by inverting the cdf; i.e., $x = F^{-1}(y)$.

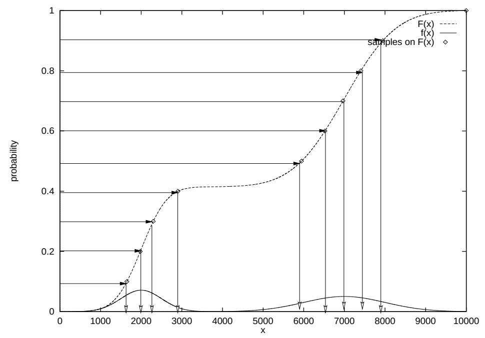


Fig. 2. Sampling using the inverse of the cdf

Let us illustrate the essence of the above rule using a simple example. Consider a function $f(x)$ as illustrated in Figure 2, which is not uniform. We generate the cdf, $F(x)$, and then take random uniformly distributed samples on the cdf (i.e.; uniformly sample the y axis). Since the cdf is uniformly distributed, the sample values capture its distribution. Now for each random sample, we calculate its inverse with respect to $F(x)$; i.e., $F^{-1}(x)$. As the figure illustrates the x values chosen essentially capture the non-uniform aspect of the underlying function $f(x)$. In particular, more samples end up being taken in the parts of the $f(x)$ that have more information.

2.4 Adapting the Golden Rule to Discrete Domains

Unfortunately the golden rule holds for a continuous probability distribution function, whereas databases in general have discrete frequency distributions. The main problem that arises, if we try to use the golden rule in our domain, is that the cdf is not a 1-to-1 mapping. In fact, the cdf in a discrete domain can be viewed as a monotonically nondecreasing step function. Consider one such step as shown in Figure 3. Several random samples in the frequency domain, e.g., c', c'', c''' all correspond to the same value in the attribute domain, namely d_{i+1}. This arises due to the fact that $F(x)$ is not a 1-to-1 function with respect to $f(x)$ in a discrete domain. The main challenge now is to determine the appropriate sample when we encounter such a situation. There are basically two possible sample pairs $\langle d_i, c_i \rangle$ and $\langle d_{i+1}, c_{i+1} \rangle$ as shown in Figure 3. If we choose $\langle d_i, c_i \rangle$, we refer to that method as *sampleLo*. Alternatively, if we choose $\langle d_{i+1}, c_{i+1} \rangle$, we call it *sampleHi*.

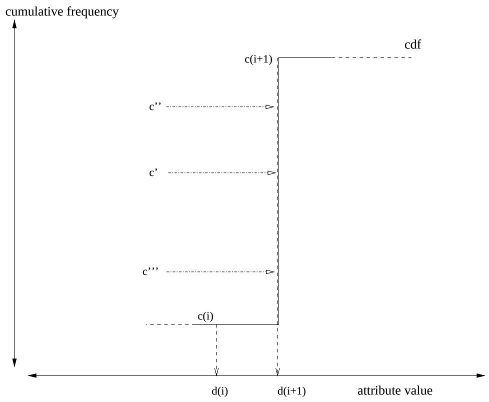


Fig. 3. Sampling in the discrete domain

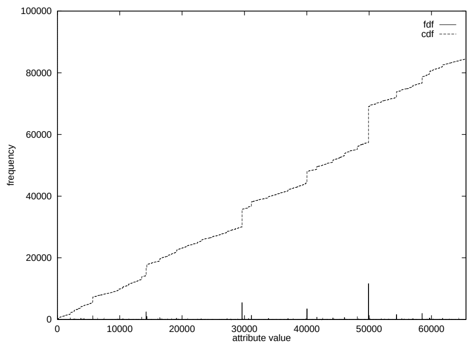
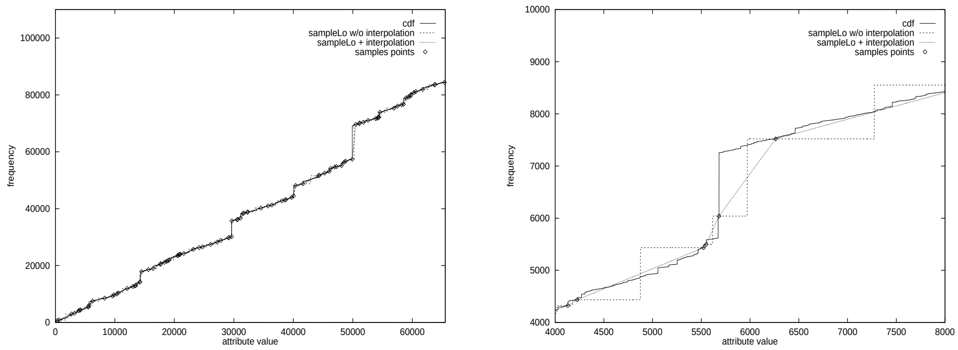


Fig. 4. fdf and cdf with Zipf distribution

We conducted a simple experiment to test the relative benefits of the different sampling approaches identified. The setting of this experiment is as follows. The domain of the attribute values is from 0 to $2^{16} - 1$. The number of tuples in the relation is 10^5. We employ the Zipf distribution in this experiment [Zip49, Poo97] with $z = 0.1$. Figure 4 shows an example fdf and cdf of one such attribute in the given relation. The vertical lines on the x-axis correspond to the frequencies associated with each attribute value and the dashed line is the resulting cumulative frequency distribution.

Figure 5.(a) shows the result of sampling the cdf with the *sampleLo* method (similar results can be obtained by using *sampleHi*). We take 100 samples from the cdf using the golden rule and depict them using the diamond shape. As discussed earlier, a query could either use the closest sample or interpolate between the two surrounding samples. We compare the approximation approaches both with and without interpolation. In Figure 5 and the subsequent figures in this section, we approximate the cdf as follows: for each attribute value, we use the samples to estimate the value of the cdf. For the purpose of the analysis of various sampling approaches, we concentrate on a detailed part of Figure 5.(a), namely the attribute range of [4000 : 8000] as shown in Figure 5.(b). The dashed line is the approximation of the cdf using the sampling rule *sampleLo*. It shows a step function because *sampleLo* uses the nearest neighbor in the sample set S_{cdf} to represent the values across the attribute value domain, hence a flat line region lies on both sides of each sample point. The dotted line represents the effect of using interpolation to answer queries based on the *sampleLo* method. In theory, the more the approximated cdf is similar to the original cdf, the more accurate the range query estimation would be. One can see how the interpolation method results in a smoother approximation, which is closer to the original cdf. However, both techniques, *sampleLo* with and without interpolation do not approximate sharp changes in the cdf very well. Recall that sharp changes are frequent in Zipf distribution. As Figure 5.(b) illustrates for the sudden change near 5700, *sampleLo* with interpolation conservatively approximates the sudden change with a gradual linear interpolation, which may give rise to higher errors.



(a) entire range (b) close up

Fig. 5. Approximation with sampleLo method with or without interpolation

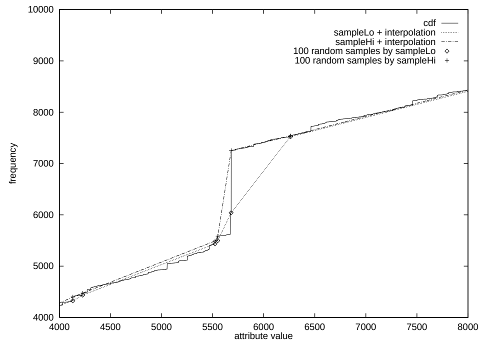


Fig. 6. Estimations using sampleLo and sampleHi sampling techniques

One reason for the large estimation error when using either the *sampleLo* or *sampleHi* method is that a single sample is taken at the sharp changing edges of the cdf. If only one sample is taken, then the estimation error between this sample and the next one is still large due to the large frequency difference. In fact, as depicted in Figure 6, neither *sampleLo* nor *sampleHi* produces good estimates if the query points fall between the sample at the sharp edge and the next neighboring sample even with interpolation.

One simple modification of the sampling method is when the first sample c' with $d_i \le d' \le d_{i+1}$ is taken, insert $\langle d_i, c_i \rangle$ into the sample set S_{cdf} as in *sampleLo*; if another sample c'' occurs in the same range with $d_i \le d'' \le d_{i+1}$, insert the other sample $\langle d_{i+1}, c_{i+1} \rangle$ into the sample set S_{cdf} as in *sampleHi* (see Figure 3). The intuition is that if the frequency f_i of d_i is large enough, there is a greater chance that the cdf range from c_i to c_{i+1} will be sampled more than once. If that is the case, we should take both $\langle d_i, c_i \rangle$ and $\langle d_{i+1}, c_{i+1} \rangle$ in order to best approximate the original cdf. However, if there is yet another sample c''' such that $d_i \le d''' \le d_{i+1}$, then since we have already taken both $\langle d_i, c_i \rangle$ and $\langle d_{i+1}, c_{i+1} \rangle$, no new information can be gained by taking the same sample again, we simply discard c''' and replace it by drawing another random sample. We refer to this modified sampling method as the *Golden Estimator*.

In Figure 7 we show the result of using the *Golden Estimator* for the same distribution used earlier. The *Golden Estimator* curve tracks the cdf curve closely. Furthermore, note how the sharp changing edge near 5700 is well approximated using *Golden Estimator*.

The complete sampling and query estimation algorithms of the *Golden Estimator* are given in Appendix B.

3 Performance Evaluation

In this section we conduct a set of experiments to compare the performance of the *Golden Estimator* with current estimation methods. We choose two of the most recent and competitive methods namely

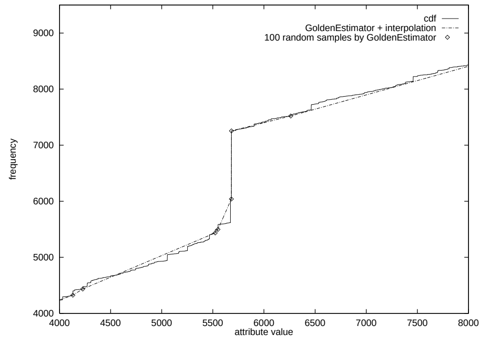


Fig. 7. Estimations using the new *Golden Estimator* techniques

the histogram method MaxDiff(V,F) proposed by Poosala et al. [PIHS96] and the wavelet based approach proposed by Matias et al. [MVW98]. We also compare the performance with the *tuple sampling* technique. In general, the tuple sampling techniques work as follows: A random sample of tuples is taken from the relation a priori or during the query processing phase. Let sample set S be taken from the relation R. When a range query Q is presented, if the subset t of S satisfied Q, the selectivity is estimated as $\frac{|R| \times |t|}{|S|}$.

3.1 Experimental Setting

We consider a relation R with 10^5 tuples. The attribute A on which range queries are specified are integer with range from 0 to $2^{16}-1$. Since the Zipf distribution has been shown to best describe data in the real world, we use the Zipf distribution for most of the experiments (except in Section 3.3). To ensure that there is no correlation between attribute values and their frequencies, we randomly assign the Zipf frequencies to the attribute values. We use the algorithm described in [Zip49] for generating the Zipf distribution.

In the experiments we explore the performance of the different methods on query sets with different selectivities. The selectivity sel of query $q(a \leq X \leq b)$ is defined as $sel = (b-a)/N$, which is the percentage of the query range to the entire attribute domain. We varied the selectivity experiment in steps of 0.01 from 0.01 to 0.8; i.e., an experiment was conducted for selectivity 1%, 2%, ..., 80% of the attribute domain. For each such selectivity, a query set of 5000 random range queries was generated. The error measure used to compare the different methods is the average relative error for the query set Q; i.e., if we assume $Q = \{q_1, q_2, ..., q_m\}$, then

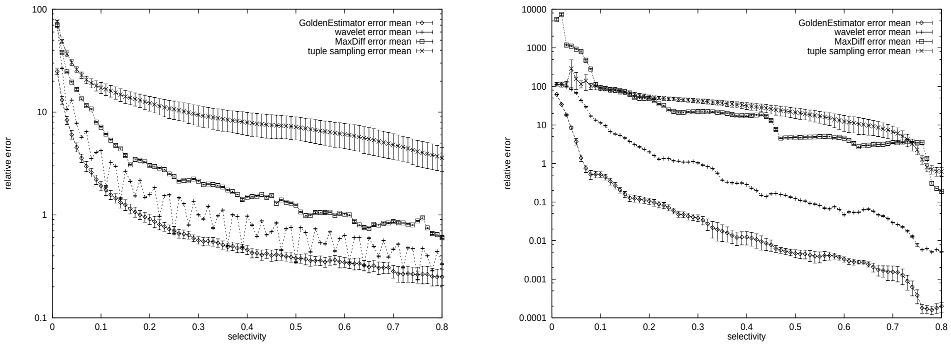
$$\overline{E}_{rel}(Q) = \frac{\sum_{i=1}^{m} E_{rel}(q_i)}{m} \tag{3}$$

A problem that has been observed [MVW98, PIHS96] with this equation is that when the actual result size is zero, the relative error E_{rel} becomes infinity. For computational reasons, in such cases, we readjust E_{rel} from infinity to 100%. Also note that the E_{rel} is not limited to 100% and can exceed 100%. This is typically the case when the actual result size s is small as in Equation 1. In all figures, the relative error is depicted on the y-axis using a logarithmic scale.

We used the independent replication method to conduct our experiments. We repeated each experiment sufficient times to obtain 90% confidence intervals, some of which are shown in the figures included in this section.

3.2 Relative Accuracy of the Golden Estimator

The first experiment compares tuple sampling, the wavelet method [VWI98], the MaxDiff histogram method [PIHS96], and the *Golden Estimator* using the same Zipf distribution; i.e., $z = 0.1$ (this represents a slightly skewed distribution where $z = 0$ refers to a uniform distribution). For each technique, we compare the performance assuming the space requirement for each are identical. We use 100 random samples for the random sampling technique, 100 wavelet coefficients for the wavelet technique, 100 histogram bins for the MaxDiff histogram technique, and 200 tuple samples for the tuple sampling technique. Note that in an actual implementation of the MaxDiff histogram, two or three numbers are actually needed for each bin to store information depending on whether there is more than one attribute value in the same bin. We choose to ignore such differences and assume that only two numbers are needed for each bin and thus assigned more bins to the MaxDiff method. Also note that for the tuple sampling technique, the sample of tuples is stored as an unordered set of domain values and thus can store up to twice as many samples as compared to the *Golden Estimator* technique.



(a) z = 0.1 (b) z = 2.0

Fig. 8. Relative error using different estimation techniques

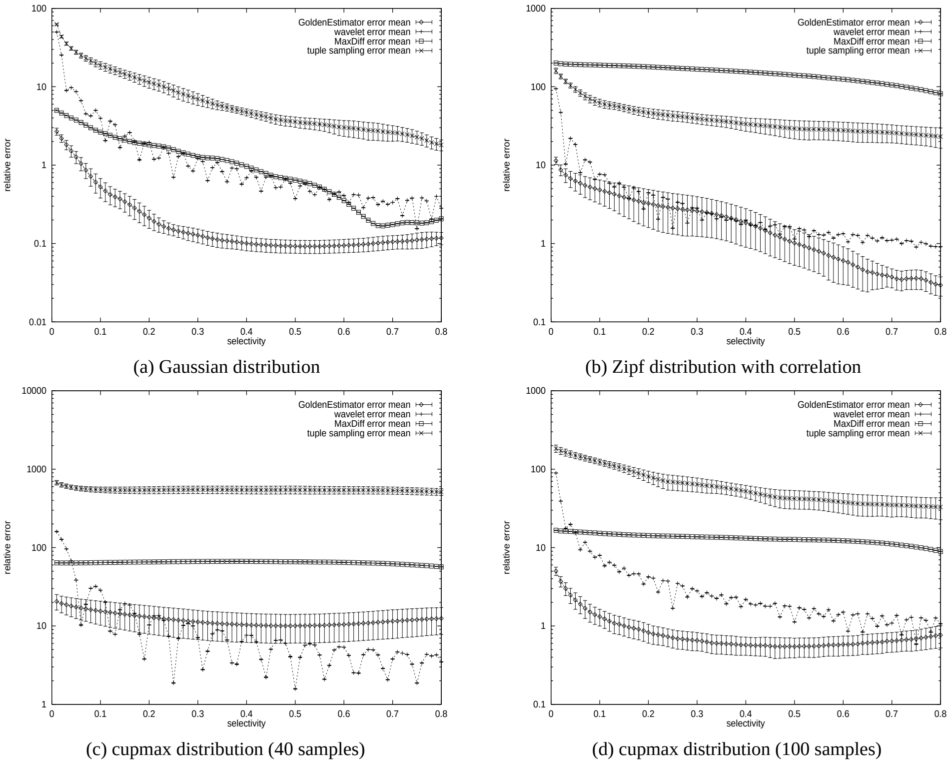
Figure 8.(a) shows the estimation error with the 90% confidence intervals. The graph clearly shows that the *Golden Estimator* results in smaller errors than the three other methods under all query selectivities. Since the confidence intervals shown are fairly tight, it also shows that taking the sample set randomly (which was done for independent replication) from the cdf has little effect on the estimation accuracy of *Golden Estimator*.

Figure 8.(b) shows the estimation error of the different methods when the underlying distribution is highly skewed. We use the same experimental setting as the previous experiment except the z value of the Zipf distribution is changed from 0.1 to 2.0; i.e., more skewed distribution. As the result shows the *Golden Estimator* still has less estimation errors than all other methods even with highly skewed data.

3.3 Comparison Using Other Data Distributions

In the previous section, we considered data with uncorrelated Zipf distribution. In this section, we explore the relative performance of the different estimation approaches using values that are correlated to each other under a variety of distributions. Such correlation may be particularly favorable for histogram based methods where the attribute domain is divided into b bins by some sort parameters under different criteria [Poo97]. Since a small amount of information is used to describe the frequencies of each attribute values in the same bin, intuitively it would seem that if the variance of values in a bin is small then the bin information will be more accurate to estimate each of the individual values. An extreme case is that all attribute values in the bin have the same frequency, then the bin keeps a single frequency value to describe all of the values without any error. The MaxDiff histogram method exploits this property by putting neighboring attribute values with large frequency differences into separate bins to reduce the average estimation error of each bin. However if neighboring attribute values have similar frequencies, histogram methods should achieve less estimation error with the same amount of storage.

The next set of experiments tests this hypothesis by comparing the performance of the MaxDiff histogram with the *Golden Estimator*, the wavelet based approach, and the tuple sampling technique. In particular, we use different data distributions where neighboring attribute values are correlated



(a) Gaussian distribution

(b) Zipf distribution with correlation

(c) cupmax distribution (40 samples)

(d) cupmax distribution (100 samples)

Fig. 9. Estimation error with different fdf distributions

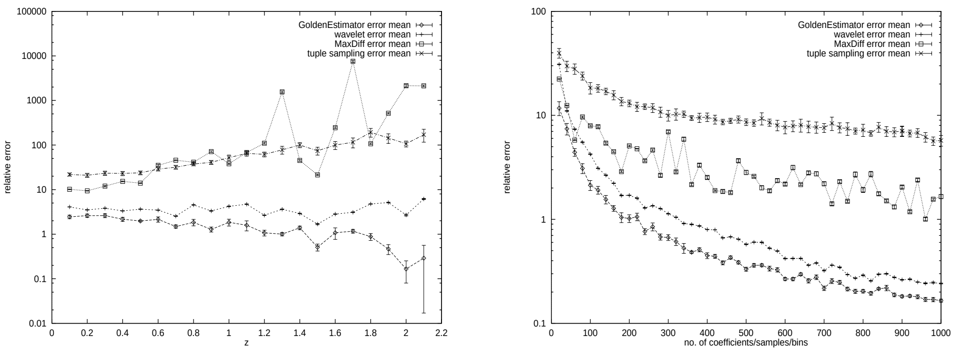
according to the distribution. Our first experiment uses a Gaussian distribution with mean equal to one half of the domain size and standard deviation equal to one fourth of the domain size (shown in Figure 13.(a) in Appendix C). Figure 9.(a) shows the result of this experiment. There are several interesting observations to make. First we note that the performance of the MaxDiff has improved when compared to the earlier uncorrelated data distributions of Section 3.2. In fact, when comparing Figure 8 and 9.(a) we note that the error has been reduced by almost an order of magnitude. This confirms our intuition that when attribute values are correlated, it is to the advantage for MaxDiff histogram method. Second we note that in fact the relative performance as compared with the result in Section 3.2, between MaxDiff and wavelet has changed. Both have relatively comparable error estimation for most of the range of selectivities. Finally, we note that the *Golden Estimator* still outperforms all other methods.

We now study two different data distribution that are particularly skewed with long correlated zero sequences. Such distributions are more appropriate for the wavelet based approach since it captures changes in data distribution in an elegant manner.

In our first such experiment we use a Zipf distribution with $z = 0.1$ and where the values are correlated according to this distribution (see Figure 13.(b) in Appendix C); i.e., smaller attribute values intentionally have higher frequencies as specified by the Zipf distribution (note that in real applications values are usually not correlated in this manner, although for some application this might be the case). The result is shown in Figure 9.(b). We first note that the performance of MaxDiff has not changed relative to the other methods. However, the error due to the wavelet based approach is now much better and in fact is comparable to the error of the *Golden Estimator* for selectivity queries in the range 0.1 to 0.5. It is, however, interesting to note that for low selectivity query, i.e., less than 4%, the *Golden Estimator* gives lower error estimation than the wavelet approach. Thus, for realistic query sizes (less then 10% selectivity), the *Golden Estimator* clearly provides better estimation.

Our next experiment uses a more extreme distribution, the so called *cupmax* distribution with correlated values. The *cupmax* distribution has been used in previous studies and is best described as a mirrored Zipf distribution with a decreasing Zipf distribution in the lower end of the attribute domain and an increasing Zipf distribution on the other end. A detailed description of the *cupmax* distribution can be found in [Poo97]. Figure 13.(c) in Appendix C shows an example fdf with *cupmax* distribution (with $z = 1.0$) using the *semi-Zipf* distribution as described in [VW99]. The domain is reduced to $[0 : 4096]$ and a relation size is set to 10^6 tuples, which is similar to the experiments in [VW99]. The storage space for each method is also reduced to 40 samples for the *Golden Estimator* method, 40 coefficients for the wavelet method, and 40 bins for the MaxDiff histogram method. Figure 9.(c) shows that the *Golden Estimator* has greater average error than the wavelet method when the selectivity is greater than 4%. For low selectivity, less than 4%, the *Golden Estimator* has lower average error. In fact for 1% and 2% selectivities (which are quite reasonable for real applications) the difference in error between the two methods is more than an order of magnitude. For higher selectivity ($\geq$ 10%), the wavelet approach outperforms the *Golden Estimator* and other techniques. The reason for this reduced performance is that in the cupmax distributed fdf, the near flat region in the central part of the attribution domain will not contribute enough samples to the sample set, while most of the queries, which are randomly chosen, will fall in this flat part of the attribute domain. With less samples for processing the range queries, the estimation accuracy suffers. But if we increase the number of samples/coefficients to 100, the estimation errors of both the *Golden Estimator* and the wavelet methods decrease as shown in Figure 9.(d), where the *Golden Estimator* outperforms all other three methods. Note that the MaxDiff histogram produces almost constant errors irrespective of the query selectivities. The reason is because the MaxDiff histogram method puts the boundaries of bins in the locations with largest frequency differences, which are localized in the two end regions of the attribute domain. There are only a few bins in the middle attribute domain, in this special case, one. Since the query set of each selectivity is randomly generated, with the individual queries evenly distributed throughout the entire domain, the only bin in the middle produces near constant errors for most of the queries, thus resulting in the near constant average errors for the MaxDiff method. The wavelet approach and the *Golden Estimator* adapt better to this flat region, especially with a larger number of samples/coefficients.

3.4 Effect of Distribution Skewedness and Storage Space



(a) skewedness (b) storage space

Fig. 10. Relative error with different skewedness and storage space

In this section we study the effect on estimation accuracy due to the skewedness of the data distribution and different storage space. We start by showing how the skewedness of the underlying data distribution affects the estimation error. We use the same setting as the experiment in Section 3.2, i.e., uncorrelated Zipf, but fix the selectivity to be 0.1; i.e., 10% selectivity. The Zipf distribution z parameter ranges from 0.1 to 2.1, which is highly skewed. The result in Figure 10.(a) shows that the *Golden Estimator* produces smaller relative error than the tuple sampling, the wavelet and the MaxDiff technique and the variance in estimation is relatively independent of skewedness of the underlying fdf.

The last experiment shows how different amounts of storage space will affect the estimation errors. We fix the selectivity to be 0.1 and $z = 0.1$. The storage space ranges from 20 to 2000 numbers, which is 10 to 1000 random samples, 10 to 1000 wavelet coefficients, 10 to 1000 MaxDiff histogram bins, or

20 to 2000 tuples for the tuple sampling techniques, respectively. Figure 10.(b) shows that the *Golden Estimator* outperforms all other three approaches for different storage requirements. It also shows that both the wavelet and the *Golden Estimator* benefit from the increase in storage resulting in the reduction of overall relative error. The MaxDiff histogram method, on the other hand, does not exhibit the same degree of advantage with respect to the increase in storage.

4 Discussion

Random sampling has long been proposed as a viable technique for query size estimation and aggregation in database management systems. Query size estimation is a critical component in DBMS for generating optimal query execution plans and other optimizations. Commercially, most DBMSs employ histogram based techniques for approximation of data distributions. In this paper we propose a new technique called the *Golden Estimator* which can be used for summarizing data distributions in database systems. The proposed technique is based on a simple observation, known as the golden rule of sampling, made by von Neumann in 1947. We adapt the golden rule of sampling, which holds for continuous probability distribution functions, for discrete domain. Unlike the histogram approach, the *Golden Estimator* is based on random sampling and therefore is highly efficient both in terms of computational and storage cost. As has been stated earlier, it is mandatory for a summarization approach to work well with different data distributions if it has to be integrated in a DBMS. This is because, a DBMS is used for a variety of applications resulting in a very wide spectrum of data that eventually populates the databases. We conducted extensive experiments to establish the validity of the *Golden Estimator* by showing that it is an effective technique for a large variety of data distributions. We also demonstrated that for realistic selectivities of range queries (less than 5% to 10%) the *Golden Estimator* can result in up to an order of magnitude reduction in estimation errors when compared to histograms [PIHS96]. We also showed that, in general, the *Golden Estimator* has better accuracy than the wavelet based technique [VWI98]. Although, it must be noted that the *Golden Estimator* has almost the same level of complexity as histograms whereas wavelet based approach requires more complex computational machinery and sophistication. In this paper, we have applied the *Golden Estimator* for range queries over a single attribute. Our future work will explore its applications for join estimations and multidimensional range queries in database systems. Also we will develop efficient techniques to address the problem of updating the fdf (based on the methods developed in [GMP97, GPA$^+$98, GM98]) and cdf (which is related to technique developed in [GAAS99]) in a dynamic environment.

References

[AD85] H. Ahrens and U. Dieter. Sequential random sampling. *ACM Transaction Mathematical Software*, 11(2):157 – 169, June 1985.

[AGPR99] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 275–286. ACM Press, 1999.

[CMN99] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasyya. On Random Sampling over Joins. In *Procedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 263–274, Philadelphia, Pennsylvania, June 1999.

[Com95] Computational Science Education Project. Introduction to Monte Carlo Methods. *http://csep1.phy.ornl.gov/mc/mc.html*, 1995.

[Eck87] Roger Eckhardt. Stan Ulam, John Von Neumann, and the Monte Carlo Method. *Los Alamos Science*, (15, Spacial Issue):135, 1987.

[GAAS99] S. Geffner, D. Agrawal, A. El Abbadi, and T. Smith. Relative Prefix Sums: And Efficient Approach for Querying Dynamic OLAP Cubes. In *Procedings of the International Conference on Data Engineering - ICDE 99*, pages 328–335, Sydney, Australia, March 1999.

[GM98] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 331–342. ACM Press, 1998.

[GMP97] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 466–475. Morgan Kaufmann, 1997.

[GPA+98] P. B. Gibbons, V. Poosala, S. Acharya, Y. Matias Y. Bartal, S. Muthukrishnan, S. Ramaswamy, and T. Suel. AQUA: System and Techniques for Approximate Query Answering. Technical report, Bell Laboratories, Murray Hill, NJ, February 1998.

[HAMS97] Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant. Range Queries in OLAP Data Cubes. In *Procedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 73–88, Arizona, 1997.

[HH99] Peter J. Haas and Joseph M. Hellerstein. Ripple Joins for Online Aggregation. In *Procedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 287–298, Phildelphia, Pennsylvania, June 1999.

[HHW97] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online Aggregation. In *Procedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 171–182, Arizona, 1997.

[HS92] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of 1992 ACM SIGMOD international conference on Management of data*, pages 341–350, 1992.

[HS95] P. Haas and A. Swami. Sampling-based Selectivity for Joins Using Augmented Frequent Value Statistics. In *Procedings of the 11th IEEE Conference on Data Engineering - ICDE 95*, pages 522–531, 1995.

[IP95] Yannis E. Ioannidis and Viswanath Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *Procedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 233–244, March 1995.

[JKM+98] H.V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala andKen Sevcik, and Torsten Suel. Optimal Histogram with Quality Guarantees. In *Procedings of the 24th VLDB Conference - VLDB 98*, New York, USA, 1998.

[LNS90] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of 1990 ACM SIGMOD international conference on Management of data*, pages 1 – 11, 1990.

[MVW98] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-Based Histograms for Selectivity Estimation. In *Procedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 448–459, Seattle, 1998.

[PI97] Viswanath Poosala and Yannis E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Procedings of the 23rd VLDB Conference - VLDB 97*, pages 486–495, Ahtens, Greece, 1997.

[PIHS96] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *Procedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, May 1996.

[Poo97] Viswanath Poosala. *Histogram-Based Estimation Techniques in Database Systems.* Ph. D. Dissertation, University of Wisconsin-Madison, 1997.

[PSC84] Gregory Piatetsky-Shapiro and Charles Connell. Accurate Estimation of the number of Tuples Satisfying a Condition. In *Proceedings of 1984 ACM SIGMOD international conference on Management of data*, pages 256–276, 1984.

[Vit85] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transaction Mathematical Software*, 11(1):37 – 57, March 1985.

[VW99] Jeffrey Scott Vitter and Min Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Procedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 193–204, Phildelphia, June 1999.

[VWI98] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. Data Cube Approximation and Histograms via Wavelets. In *Proceedings of Seventh International Conference on Information and Knowledge Management - CIKM'98*, pages 96–104, Bethesda, Maryland, November 1998.

[Zip49] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Reading, MA, 1949.

Appendix A: Mathematics for The Golden Rule for Random Sampling

By using some transformations of the fdf, it it possible to get the random sample sets described above in a cheaper way. Let us assume that there exists a transformation monotone increasing $y(x)$, which results in $dy/dx > 0$ for all x. In reality, the transformation must conserve probability of s; i.e., the probability of random variable x' occurring in dx about x must be the same as the probability of the random variable y' occurring in dy about y. More precisely, there must be a one-to-one relationship between x and y and y must exist. By the definition of *probability distribution function* (pdf), $f(x)$ and $g(y)$ are

$$f(x)dx = prob(x \leq x' \leq x + dx) \tag{4}$$
$$f(y)dy = prob(y \leq y' \leq y + dy) \tag{5}$$

The transformation implies that these probabilities must be equal, thus we have

$$f(x)dx = g(y)dy \tag{6}$$

and we can solve $g(y)$:

$$g(y) = f(x)/[dy/dx] \tag{7}$$

One of the most important transformations that satisfies the above condition is the *cumulative distribution function* (*cdf*):

$$y(x) = F(x) = \int_{-\infty}^{\infty} f(x')dx' \tag{8}$$

In this case, we have $dx/dy = f(x)$ and more importantly,

$$g(y) = 1, 0 \leq y \leq 1 \tag{9}$$

Equation 9 shows that the cdf is uniformly distributed on [0,1], independently of the pdf $f(x)$. This result has significant impact on how we can sample from an arbitrary pdf. Because the random variable x and the cdf $F(x)$ are one-to-one correspondent, we can sample x by first sampling $y = F(x)$ and then use the reverse function to invert $F(x)$ to x, or $x = F^{-1}(y)$. And from Equation 9 we know the cdf is uniformly distributed on [0,1]. Therefore, by using a uniform random number generator $U[0,1]$ to generate a sample ξ from cdf $F(x)$ we can have a sample about x by $x = F^{-1}(\xi)$. The result is as shown in Figure 2.

Appendix B: The *Golden Estimator* Sampling and Query Estimation Algorithm

Figure 11 depicts the algorithm that is used for drawing random samples based on the *Golden Estimator* method. The function has two input parameters: number of samples and the cdf of the target attribute value tuples in a relation. The main loop of the algorithm generates a random number s between 0 and the maximum value of the cdf. We then identify $\langle x_i, c_i \rangle$ in the cdf such that x_i is the largest value for which $s \geq c_i$ and $\langle x_{i+1}, c_{i+1} \rangle$ such that x_{i+1} is the smallest value for which $s \leq c_{i+1}$. Depending upon the samples accumulated so far, first $\langle x_i, c_i \rangle$ is included and if that already exist and $\langle x_{i+1}, c_{i+1} \rangle$ is distinct then $\langle x_{i+1}, c_{i+1} \rangle$ is added to the sample set. The algorithm terminates after the specified number of samples are drawn.

Figure 12 illustrates the algorithm that is used to estimate the answer for a given range query $[a : b]$ from the set of random samples. As shown, for each end point a and b we find the corresponding estimates for cdf as c'_a and c'_b. c'_a is the interpolated value from the nearest random samples that lie on either side of a in the random sample set. Similarly for c'_b. The query result is estimated to be the difference $c'_b - c'_a$.

Appendix C: Example Data Distributions

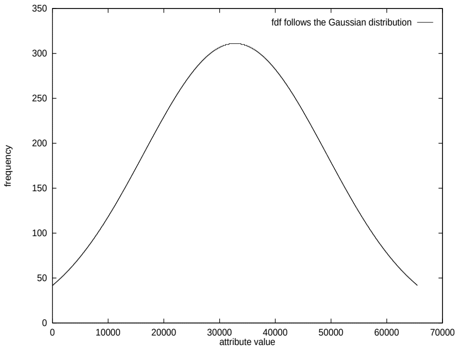
Figure 13 depicts the frequency distributions of the synthesized data used in the experiments in Section 3.3.

```
PROCEDURE GoldenEstimator (noOfSampleNeeded, cdf)
BEGIN
    sampleSet = NULL; (* keep samples *)
    sampleCount = 0; (* keep track of the number of samples taken *)
    s = 0; (* random sample *)
    xLo = 0; (* low attribute value *)
    xHi = 0; (* high attribute value *)
    WHILE sampleCount ≤ noOfSampleNeeded
        s = pick a random number between 0 and c_{n-1} in cdf;
        xLo = Max(x_i), where c_i ≤ s, 0 ≤ i ≤ n − 1;
        xHi = Min(x_i), where c_i ≥ s, 0 ≤ i ≤ n − 1;
        IF ⟨xLo, c_{xLo}⟩ ∉ sampleSet THEN
            add ⟨xLo, c_{xLo}⟩ to sampleSet;
            sampleCount + +;
        ELSE
            IF ⟨xHi, c_{xHi}⟩ ∉ sampleSet THEN
                add ⟨xHi, c_{xHi}⟩ to sampleSet;
                sampleCount + +;
            END; (* IF *)
        END; (* IF *)
    END; (* WHILE *)
    RETURN sampleSet;
END; (* GoldenEstimator *)
```

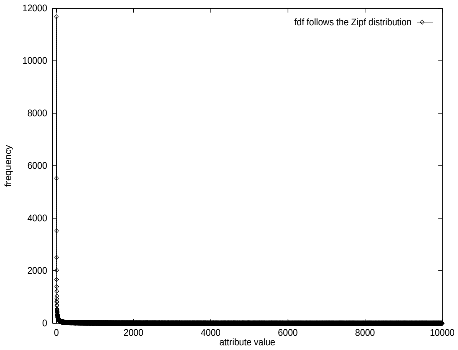
Fig. 11. *Golden Estimator* Sampling Algorithm

```
PROCEDURE queryEstimate (a, b, sampleSet) (* query = q(a ≤ X ≤ b) *)
BEGIN
    c'_a, c'_b; (* estimates of c_a and c_b *)
    dLo, dHi; (* low/high attribute value *)
    cLo, cHi; (* low/high cumulative value *)
    (* find nearest samples that lies on either side of a*)
    cLo = Max(c_i), dLo = Max(d_i), where d_i ≤ a, ⟨d_i, c_i⟩ ∈ sampleSet;
    cHi = Min(c_i), dHi = Min(d_i), where d_i ≥ a, ⟨d_i, c_i⟩ ∈ sampleSet;
    (* interpolate c_a using Eq 2 *)
    c'_a = clo + (cHi − Clo) * (a − dLo)/(dHi − dLo);
    (* similarly for b*)
    cLo = Max(c_i), dLo = Max(d_i), where d_i ≤ b, ⟨d_i, c_i⟩ ∈ sampleSet;
    cHi = Min(c_i), dHi = Min(d_i), where d_i ≥ b, ⟨d_i, c_i⟩ ∈ sampleSet;
    c'_b = clo + (cHi − Clo) * (b − dLo)/(dHi − dLo);
    RETURN c'_b − c'_a;
END; (* queryEstimate *)
```

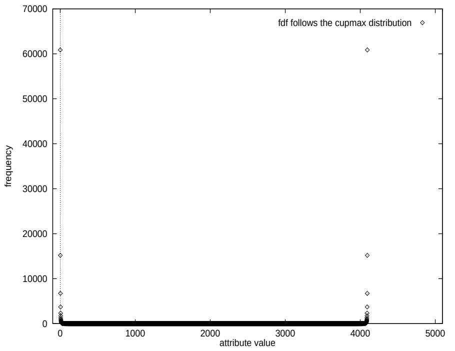
Fig. 12. The *Golden Estimator* Query Estimation Algorithm



(a) Gaussian distribution



(b) Zipf distribution with correlation (z = 0.1)



(c) cupmax distribution

Fig. 13. Example Data Distributions