Random Sampling for Histogram Construction: How much is enough?

Surajit Chaudhuri
Microsoft Research
E-mail: surajitc@microsoft.com

Rajeev Motwani
Stanford University
E-mail: rajeev@cs.stanford.edu

Vivek Narasayya
Microsoft Research
E-mail: viveknar@microsoft.com

Abstract

Random sampling is a standard technique for constructing (approximate) histograms for query optimization. However, any real implementation in commercial products requires solving the hard problem of determining *"How much sampling is enough?"* We address this critical question in the context of equi-height histograms used in many commercial products, including Microsoft SQL Server. We introduce a *conservative* error metric capturing the intuition that for an approximate histogram to have low error, the error must be small in *all* regions of the histogram. We then present a result establishing an optimal bound on the amount of sampling required for pre-specified error bounds. We also describe an *adaptive page sampling* algorithm which achieves greater efficiency by using all values in a sampled page but adjusts the amount of sampling depending on clustering of values in pages. Next, we establish that the problem of estimating the number of distinct values is *provably difficult*, but propose a new error metric which has a reliable estimator and can still be exploited by query optimizers to influence the choice of execution plans. The algorithm for histogram construction was prototyped on Microsoft SQL Server 7.0 and we present experimental results showing that the adaptive algorithm accurately approximates the true histogram over different data distributions.

1 Introduction

The emerging interest in decision support systems has reinforced the importance of processing complex queries efficiently over *large* data warehouses. Efficient processing of such queries requires not only a high-performance query processing system, but also needs an optimizer that is able to make a judicious choice of an execution plan. The ability of an optimizer to make a good decision is critically influenced by the availability of statistical information on tables referenced in queries. Such statistical information is automatically collected for columns of tables over which indexes are constructed during the index generation phase. However, statistics only on indexed columns are not sufficient. In a database system, the ability to obtain statistical information *efficiently and accurately* on columns that do not have indexes is of great importance. In addition to greatly enhancing the accuracy of the optimizer's estimates, such statistical information is of great interest in tools that help decide physical database design [7, 6].

In many commercial systems, including Microsoft SQL Server, the optimizer relies on *histograms* on selected columns to estimate selectivities of queries. In addition to histograms, *density* as well as number of *distinct values* in the columns are other statistical parameters of interest. Generating histograms and other statistical measures over large data sets is an expensive proposition. Therefore, random sampling of the data has been considered as a technique to efficiently construct approximate histograms. However, the ability to successfully use random sampling also requires that we be able to solve the problem of determining the *degree of sampling necessary for the desired accuracy*. Several practical considerations make this problem challenging:

- Need small error for all queries, with high probability,
- Data distribution is not known a priori.
- Columns have an unknown number of duplicate values.
- For reasonably large samples, sampling at the tuple level is prohibitively expensive, but tuples within a disk block may be correlated.

We present analytical results and extensive experimental results on Microsoft SQL Server 7.0 that provide a basis for a practical and accurate technique for constructing histograms via random sampling. Our contributions are:

(a) We introduce a new error metric to measure the "local" differences between the perfect histogram (obtained by a full scan and sorting of the given column) and an approximate histogram obtained via random sampling. The local nature of the error metric helps us guard against unacceptably large errors in estimation due to differences between the approximate and the perfect histograms, unlike earlier work based on more "global" error metrics.

(b) We establish an optimal *bound* on the amount of sampling needed for a prespecified error bound without making *any* assumptions about data distribution.

(c) We present a page-level sampling algorithm that adapts to correlation among data tuples that reside on the same page. This enables us to harness the efficiency inherent in page level sampling without sacrificing accuracy should data in blocks happen to be correlated.

(d) We discuss extensions for handling duplicate values.

(e) We show that the specific problem of estimating distinct values is provably difficult by establishing a limit on the achievable accuracy of estimation in the worst-case. We devise a simple estimator which we believe is optimal.

In this paper, we focus on equi-height (or, equi-depth) histograms. We chose equi-height histograms for our analysis since they are commonly used in many commercial optimizers, including SQL Server. In the full paper [5], we also discuss how the above analysis can be adapted to the related family of compressed histograms. Extending our results to the case of other histogram structures [15, 16] is one of our ongoing research goals.

The rest of the paper is organized as follows. In the remainder of this section, we briefly summarize past work in the area. In Section 2, we introduce equi-height histograms and describe our proposal for the error metric to measure the accuracy of a histogram based on sampling; the power of this approach is illustrated via an application of range query output size estimation. An optimal bound on the amount of sampling needed for a given error bound is presented in Section 3. The adaptive sampling algorithm for block sampling is discussed in Section 4. Section 5 discusses how the error metric and the results have a natural generalization when the data values have duplicates. The issue of distinct values is discussed in Section 6. Section 7 describes extensive experiments on Microsoft SQL Server that provides an experimental analysis of the results presented in this paper; in particular, we study the effect of correlation within disk blocks on sampling requirements. We conclude this section with a summary of our proposal for effective use of sampling for approximating histograms and distinct values.

1.1 Related Work

Random sampling has been proposed and used in many different contexts in databases [23, 24]. In particular, a large body of work [18, 19, 20, 13, 14, 12, 11, 17] has addressed the problem of estimating result size for a given query using random sampling. In contrast, the problem that we have addressed requires us to estimate a histogram using random sampling. Unlike result size of a specific query, a histogram is a complex structure having many variables (bin boundaries) that need to be estimated. Furthermore, in most earlier work, the recommended sampling bounds are based on distribution-specific assumptions and heuristic analysis.

The use of random sampling for estimating histogram was proposed by Piatetsky-Shapiro and Connell [27]. They show that *given a particular query*, only a small sample size is needed to estimate a histogram that is adequate for the query with a high probability. In sharp contrast, the problem of approximating histograms require us to derive a histogram that is reasonably accurate for a large class of queries (preferably all) with high probability. Although the main focus of the work in Gibbons, Matias, and Poosala [8] is on the incremental maintenance of histograms, they do provide a distribution-independent bound on the required amount of sampling. However, our bounds are significantly stronger and lends to ease of use. We will discuss the relationship between this and our results in Section 3. As regards the problem of estimating the number of distinct values, we have sharpened the intuition obtained in earlier work [10, 26] by providing a negative result which explains why distinct values cannot be approximated reliably. In addition, we present a new estimator that is provably optimal and performs extremely well in our experiments.

2 A New Error Metric for Equi-Height Histograms

In this section we review the definition of an equi-height histogram and approximate histograms. We show that using the standard error metrics for approximate histograms can be extremely misleading and could induce large errors in query cost estimation for simple range queries. We then introduce our new metric for measuring the error in an approximate histogram and show that it overcomes the problems with earlier metrics.

2.1 Equi-Height and Approximate Histograms

Suppose we have a relation with n tuples, containing an attribute X defined over a domain $\mathcal{D}$. The *value set* $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of values for this attribute X in the n tuples, i.e., $v_i \in \mathcal{D}$ is the value of X in the ith tuple of the relation. We assume that $\mathcal{D}$ is totally ordered.

A *k-histogram* for a value set V is a partition of $\mathcal{D}$ into k intervals defined by a sequence of *separators* $s_1, s_2, \ldots, s_{k-1} \in Z$. This induces a partition of V into the k buckets $B_1, B_2, \ldots, B_k$ such that for all j, $1 \leq j \leq k$, $B_j = \{v_i \in V \mid s_{j-1} < v_i \leq s_j\}$, where we define $s_0 = -\infty$ and $s_k = +\infty$ for convenience. We denote by $b_j = |B_j|$ the size of the bucket B_j. A k-histogram for V is said to be an *equi-height k-histogram* if each bucket size, b_j, is exactly n/k.

Note that an exact equi-height k-histogram may not even exist, specifically when V contains repeated values (i.e., is a multiset). We will examine the issue of duplicate values in greater detail a bit later (Section 5). For now, let us assume that V does not contain any duplicated values and that a "perfect" equi-height k-histogram for V exists.

A major emphasis of this paper is the computation of histograms for large disk-resident databases by taking random samples to avoid scanning the entire relation. We cannot expect to be able to compute a *perfect* equi-height histogram from a random sample. Consequently, we focus on finding a k-histogram that comes as close as possible to ensuring that each of the k buckets is of size n/k, i.e., an *approximate* equi-height k-histogram. It becomes important to define a notion of error to measure how well the perfect histogram is approximated by the one that is computed from a sample. We now turn to a discussion of such metrics.

2.2 A Critique of Standard Error Metrics

In the literature, the error in approximate histograms with respect to perfect equi-height histograms is usually measured in terms of the *average error* Δ_{avg} and *variance error* Δ_{var}, which are formally defined as follows:

$$\Delta\text{avg} = \sum_{1 \leq j \leq k} \frac{\left|b_j - \frac{n}{k}\right|}{k}, \quad \text{and} \quad \Delta\text{var} = \sqrt{\sum_{1 \leq j \leq k} \frac{\left|b_j - \frac{n}{k}\right|^2}{k}}.$$

These error metrics measure the mean and the variance of the absolute differences of the bucket sizes in the approximate and the perfect histograms.

To illustrate the problem with these error metrics, let us consider the use of histograms with *seemingly* small average and variance errors for estimating the output size for range queries. Suppose that we have histograms where either the average error or variance error is bounded by $\delta = fn/k$, where f measures the error as a fraction of the ideal bucket size n/k. Consider a range query which specifies an interval $I = [x, y]$ in the domain $\mathcal{D}$ of an attribute X. The following

is a typical strategy[1] for estimating the number of tuples in which the attribute X has a value in the range I. Given the range $[x, y]$, we first locate the buckets B_a and B_b containing the values x and y, respectively. For $a < b$, we count the total number of values in the buckets $B_{a+1}, \ldots, B_{b-1}$ and add to these the interpolated number of values larger than x in B_a and smaller than y in B_b. The interpolation requires some assumptions about the distribution of values inside a bucket and is the main source of error in the estimation.

We can measure the estimation error for a query as either the absolute or the relative difference from the correct output size, but in the latter case we need to assume that the output size is not too small to get any meaningful numbers. In either case, Theorem 1 say that the average and variance error bounds do not allow any reasonable guarantees on the error for even the simple range queries; certainly, the estimation errors are far worse than would have been the case with a *perfect* histogram[2]. The proof is in the full paper [5].

Theorem 1 *Consider a range query with output size $s = tn/k$ for some $t < 1$. (All bounds below are tight.)*

1. *A* perfect equi-height *histograms cannot guarantee absolute error $\alpha < \frac{2n}{k}$ or relative error $\beta < \frac{2}{t}$ for all range queries.*
2. *A histogram with* average error $\Delta_{avg} = fn/k$ *cannot guarantee absolute error $\alpha < \left(1 + \frac{fk}{4}\right)\frac{2n}{k}$ or relative error $\beta < \left(1 + \frac{fk}{4}\right)\frac{2}{t}$ for all range queries.*
3. *A histogram with* variance error $\Delta_{var} = fn/k$ *cannot guarantee absolute error $\alpha < \left(1 + f\sqrt{\frac{kt}{8}}\right)\frac{2n}{k}$ or relative error $\beta < \left(1 + f\sqrt{\frac{kt}{8}}\right)\frac{2}{t}$ for all range queries.*

Example 1 *Let us consider a numerical example based on some typical numbers. Suppose that $k = 1000$ and $f = 0.05$ (i.e., the histograms have 5% error). Consider a range query with output size $s = 10n/k$, i.e., $t = 10$. Then, the perfect histogram gives absolute error $\alpha = 0.002 \times n$ and relative error* 0.2 *in estimating s. However, when using a histogram with average error bounded by fn/k,*

the estimation errors go up by a multiplicative *factor of* 13.5. *For histograms with variance error bounded by fn/k, the situation is slightly better in that the estimation error goes up by a* multiplicative *factor of* 2.8*; however, increasing the value of s will further increase the error in the latter case.*

2.3 The Max Error Metric

We propose a new error measure for approximate histograms.

Definition 1 *The* max error *metric for a k-histogram is defined as follows:*

$$\Delta_{max} = \max_{1 \le j \le k} \left| b_j - \frac{n}{k} \right|.$$

A k-histogram with $\Delta_{max} \le \delta$ is said to be a δ-deviant histogram, and satisfies the property that for all j, $1 \le j \le k$,

$$\left| b_j - \frac{n}{k} \right| \le \delta.$$

[1] We are assuming that no additional information is stored along with the histogram other than the count of the number of values inside each bucket; our discussion below will have to be suitably modified if there is extra information available.

[2] Note that even the perfect histogram cannot give zero estimation error since it is only a summary of the data.

We are mainly interested in the case $\delta = fn/k$ for $0 < f < 1$. It is important to realize that our notion of error for an approximate equi-height k-histogram is far stronger than the measures of error considered earlier in the literature. We require that *every* bucket in the k-histogram has a size which has a *small absolute* difference with respect to the bucket sizes in an exact equi-height k-histogram. That is, the approximate histogram must accurately reflect the perfect histogram *locally* at all points. Further, our goal is to achieve a small value of f; that is, δ is not merely small relative to the total data size n, but it is required to be a small fraction of bucket size (n/k). Thus, *for each bucket*, we strive to keep the error to a *small fraction of the bucket size*.

In contrast, the average and variance error metrics are aggregates of errors in bucket sizes which only give a *global* error over the histogram. Bounding these aggregates does not rule out the possibility that some bucket had extremely large error in its size, which is not the case with our metric. We believe that the max error metric is far better than the other two for query cost estimation since it attempts to more tightly bound the error that will occur in cost estimation. We will illustrate the power of this definition by applying it to the problem of output size estimation for range queries. First, we contrast the various notions of error.

Example 2 *Suppose we have a histogram with $k = 10$ buckets of the following sizes: 88, 101, 87, 88, 89, 180, 90, 88, 103, 86. Note that $n = 1000$ and thus the perfect histogram has* 10 *buckets of size* 100 *each. Consider now the various measures of error as applied to this situation: average error $\Delta_{avg} = 16.8$; variance error $\Delta_{var} = 27.5$; and, max error $\Delta_{max} = 80.0$. As the value of k increase, the gap between the various notions of error can increase unboundedly.*

Theorem 2 shows that if a k-histogram has max error bounded by δ (i.e., is δ-deviant) then its average and variance error is also bounded by δ. The converse is obviously not true, implying that *the max error is the strongest notion of error.* The proof is in the full paper [5].

Theorem 2 *If a k-histogram has max error $\Delta_{max} \le \delta$ (or, is δ-deviant) then it must be the case that: $\Delta_{avg} \le \delta$, and $\Delta_{var} \le \delta$. However, the converse is not true in general.*

We illustrate the power of the max error metric by applying it to the output size estimation for range queries. Theorem 3 shows that bounding the histogram error in terms of max error does indeed guarantee small estimation error for range query sizes, in fact the error is fairly close to best possible (that which is obtained by the perfect histogram). The proof is in the full paper [5].

Theorem 3 *Consider a range query with output size $s = tn/k$. An approximate histogram with max error $\Delta_{max} = fn/k$ guarantees absolute error $\alpha \le (1 + f)\frac{2n}{k}$ and relative error $\beta \le (1 + f)\frac{2}{t}$ for all possible range queries.*

Example 2 (continued) *Consider again the setting of Example 2. The perfect histogram gives absolute error $\alpha = 0.002 \times n$ and relative error* 0.2 *in estimating s, while histograms with bounded average/variance errors had estimation errors that were worse by multiplicative factors of* 13.5 *and* 2.8*, respectively. In contrast, the histogram with bounded max error has absolute error* $0.0021 \times n$ *and relative error* 0.21*, which is off by a factor of* 1.05 *from the optimal.*

While we have a more conservative and a stronger error metric, it is not clear a priori that it is possible to efficiently construct approximate histograms with small error using a

reasonable amount of sampling. Perhaps surprisingly, we will show in the next section that not only is it possible to achieve such a construction, but in fact we can provide *far stronger guarantees* on the size of the max error than known in earlier literature for the *weaker notions of error.*

3 Record-Level Sampling

We begin by our study of histogram construction by considering the model where we sample individual tuples/records uniformly at random from a relation, without taking into account the issue of disk blocking factors. This record-level sampling model considered in this section is not realistic, but yet a useful starting point as it sets limits on the quality of performance of the more realistic block-level sampling strategy to be considered in the next section, and it also allows us to develop tools that will later be helpful in our work on block-level sampling,

While sampling-based construction of histograms is well-studied in the database literature, we are not aware of any strong results specifying the inherent trade-off between the amount of sampling performed and the resulting error in the histogram (See Section 3.4). The main result of this section is an essentially optimal bound on the trade-off between these two and other related performance measures. We also show that using a slightly larger amount of sampling allows us to establish bounds on an *even stronger* error metric for histograms that may prove useful in query cost estimation. We evaluate the implications of our bounds and perform a comparison with past work along these lines.

3.1 Sampling Methodology

It is convenient to assume that we perform random sampling *with replacement* and we will adopt that approach. Of course, in practice it might be a bit more efficient to perform random sampling *without replacement.* As in most applications of random sampling, our results do carry over to the latter style of sampling without any noticeable change in the bounds obtained, but we defer the details to the full paper [5]. For sampling without replacement, the analysis involves a study of the hypergeometric distribution rather than the simpler binomial distribution.[3]

We choose a random sample $R \subset V$, such that $|R| = r$, by repeatedly picking a random value from V until we have a total of r values, without regard for repeated values. Then, we compute an equi-height k-histogram for R. Let $s_1, s_2, \ldots, s_{k-1}$ be a sequence of separators that define the equi-height k-histogram for R computed in this fashion. Finally, we compute a k-histogram for V using exactly the same sequence of separators. Let $B_1, B_2, \ldots, B_k$ be the k buckets induced in the histogram for V. The sizes of these k buckets are random variables, and it is unlikely that they are all equal. However, since the separators $s_1, s_2, \ldots, s_{k-1}$ induce an equi-height k-histogram for R, we expect that the buckets $B_1, B_2, \ldots, B_k$ are nearly equal.

3.2 Trading-Off Error and Sample Size

Now we bound the number of random samples required to guarantee that the resulting k-histogram is δ-deviant. We give essentially optimal formulas describing the trade-off between the various parameters, notably the number of buckets k, the error δ, and the number of random samples r.

[3] Refer to Motwani and Raghavan [21] for all basic definitions/results on randomization used in this paper.

Theorem 4 *Let $\delta \leq \frac{n}{k}$. An equi-height k-histogram for a random sample R of size r from a value set V of size n gives a δ-deviant k-histogram for V with probability at least $1-\gamma$ ($\gamma > 0$), provided that*

$$r \geq \frac{4n^2 \ln(2n/\gamma)}{k\delta^2} \quad \textit{or, equivalently,} \quad \delta \geq \sqrt{\frac{4n^2 \ln(2n/\gamma)}{rk}}.$$

We defer the detailed proof to the full paper [5].

We delay a discussion of the implications of Theorem 4 until after the presentation of the stronger error bound in Theorem 5 below. Note that the condition that $\delta \leq \frac{n}{k}$ is not restrictive since it is unlikely that there will be any interest in estimating the number of samples required for ensuring an error that exceeds 100%; in any case, we can easily modify our proofs to handle the more general case except that the bounds will be marginally different.

A Stronger Error Metric. We now show that it is possible to derive an even stronger bound on the extent to which the approximate and perfect histograms differ. The idea is to require a bound of δ on the difference in the precise location of the separators for the buckets, rather than merely the buckets sizes as in the max error metric in Theorem 4.

Definition 2 *Let H and H^* be two k-histograms for V with buckets $B_1, \ldots, B_k$ and $B_1^*, \ldots, B_k^*$, respectively. Then, H and H^* are said to be* δ-separated *if for all j, $1 \leq j \leq k$, it is the case that the symmetric difference of the buckets B_j and B_j^* is of size at most δ.*

The following theorem shows that the amount of random sampling needed to guarantee δ-separation is not much more than that required for ensuring δ-deviation. We defer the proof to the full paper [5].

Theorem 5 *Let $\delta \leq \frac{n}{k}$ and $\gamma > 0$. An equi-height k-histogram for a random sample R of size r from a value set V of size n is δ-separated from the perfect k-histogram for V with probability at least $1-\gamma$, provided that*

$$r \geq \frac{12n^2 \ln(2k/\gamma)}{\delta^2} \quad \textit{or, equivalently,} \quad \delta \geq \sqrt{\frac{12n^2 \ln(2k/\gamma)}{r}}.$$

3.3 Evaluating the Bounds and Their Implications

Before exploring the implications of Theorems 4 and 5, we make the bounds therein more transparent by expressing them slightly differently by setting $\delta = fn/k$ for some fraction $f \leq 1$. The value f measures the *relative* deviation from the *optimal* bucket size of n/k.

Corollary 1 *Let $\delta = f \times \frac{n}{k}$, $0 < f < 1$, and $\gamma > 0$. An equi-height k-histogram for a random sample R of size r from a value set V of size n gives a δ-deviant k-histogram for V with probability at least $1-\gamma$, provided that*

$$r \geq \frac{4k \ln(2n/\gamma)}{f^2} \quad \textit{or, equivalently,} \quad \delta \geq \sqrt{\frac{4k \ln(2n/\gamma)}{r}}.$$

Let us now evaluate the trade-off we give between the values of k, δ, and r, especially as encapsulated in Corollary 1. We focus on determining the sample size as a function of the other parameters, but we can also use our results to estimate the error for fixed values of the other parameters, or to estimate the number of buckets needed in the histogram as a function of the other parameters. This multi-functionality is

a pleasant consequence of having such an explicit and formal trade-off and should have considerable value in practice.

First of all, observe that the sample size required grows linearly in k and inversely with the squared deviation, but is *essentially independent of* n since the logarithmic dependence is negligible. This important observation is rather counter-intuitive at first. It implies that once the database size is large enough there is great value in performing random sampling, and that the random sampling cost remains fixed as the size of the database increases beyond this threshold. A similar comment applies to the dependence on γ — to get higher confidence that the random sample provides the desired error bound requires little additional sampling. For example, we can we can even choose $\gamma = 2/n$, thereby changing the logarithmic term to $\ln n^2$ and at most doubling the sample size while ensuring that the probability of exceeding the promised error bound is negligible.

Example 3 *We illustrate the trade-off results, particularly Corollary 1, via a numerical evaluations. Throughout, we will assume that error probability is* $\gamma = 0.01$. *Even for* n *as large as 1Gig, we obtain that* $\ln 2n/\gamma$ *is roughly 20.*

- **Determining Sample Size:** *For* $k = 500$ *and relative error* $f = 0.2$, *we require sample size roughly 1Meg for essentially all reasonable values of* n. *On the other hand, for* $k = 100$ *and relative error* $f = 0.1$, *we require sample size roughly 800K for all reasonable values of* n. *Such analysis allows a system to quickly pick the sample size that is appropriate for the application at hand, or to decide that it may not be cost effective to use random sampling for desired histogram size/error.*

- **Determining Histogram Size:** *Suppose we decide that we wish to sample at most 1Meg from a data set of size* n *equal to 20Meg and want deviation at most* $0.25 \times n/k$ *(i.e.,* $f = 0.25$*). What is the maximum size of a histogram that we can support? It is easy to determine that we should not have* k *exceeding 800.*

- **Determining Histogram Error:** *Finally, suppose we decide that we wish to sample 800K from a dataset of size 25Meg to create a histogram with* $k = 200$ *buckets. How much error should we expect in the histogram? The answer is that* f *is bounded by 14%.*

3.4 Comparison to Previous Work

Perhaps the most similar earlier work is that of Gibbons, Matias, and Poosala [8]. While the focus of their work is on the incremental maintenance of approximate histograms as tuples are added to or deleted from a relation, this is the only piece of work in the literature that appears to provide guarantees on error bounds for histograms that is in the same spirit as ours.

Theorem 6 ([8]) *Let* $k \geq 3$, $c \geq 4$, *and* $f = \left(c \ln^2 k\right)^{-1/6}$. *Then, with probability* $1 - \gamma$, *for* $\gamma = k^{1-\sqrt{c}} + n^{-1/3}$, *an approximate histogram with error* $\Delta_{var} = f\frac{n}{k}$ *is obtained from a random sample of size* $r \geq ck \ln^2 k$, *when* $n \geq k^3$.

While Theorem 6 was an important step in being the first result providing a formal guarantee on histogram error, our results are significantly stronger in many respects, besides being multi-functional as shown in Example 3. We highlight some of the main differences in next example.

Example 4

1. Theorem 6 only considers the *variance error*, while our notion of error is the much more conservative *maximum error. In fact, as per Theorem 2, our results also* simultaneously *guarantee the same bound on the variance error and* average error.

2. While our results are essentially independent of n, Theorem 6 applies only to extremely large values of n which are unlikely to occur in practice. *Theorem 6 requires that* $N \geq r^3$, *where* $r \geq 4k \ln^2 k$. *For even a reasonably small value of* $k = 100$, *this requires that* $n \geq 6 \times 10^{11}$ *which is almost a tera-byte of data. For* $k = 1000$, *they would require that* $n \geq 7 \times 10^{15}$.

3. Our results provide a smooth tradeoff between the key parameters k, r, and γ, whereas their results only allow one possible setting for each of these parameters. *While it is possible to obtain some kind of a trade-off from Theorem 6 by varying the value of* c, *the bounds are such that using a value of* c *significantly greater than 4 is totally impractical.*

4. Our results allow the deviation error to be reduced arbitrarily by smoothly increasing r or decreasing k, Theorem 6 does not really permit a value of f below 0.35 for any practical choice of the other parameters. *For* $k = 100$, *it guarantees* $f = 0.48$*; increasing the value of* k *doesn't help since to obtain a value of* f *smaller than 0.35 requires* $k > 100,000$. *This is because for* $f = 0.1$, *Theorem 6 requires* $k > e^{500}$ *and* $n > e^{1500}$, *and for* $k = 0.2$ *it needs* $k > e^{60}$ *and* $n > e^{180}$. *In contrast, we can easily guarantee* $f = 0.1$, *or even smaller, for reasonable values of* k.

5. Perhaps most significantly, our results guarantee a much smaller requirement of the size of the random sample than can be inferred from Theorem 6. *Setting the value of* γ *to the error probability of error given in Theorem 6, we obtain the following comparison: suppose we decide that* $k = 500$ *and* $f = 0.2$, *then we can guarantee that a sample of size 4Meg will suffice for all reasonable values of* n*; in contrast, Theorem 6 cannot possibly guarantee* $f = 0.1$, *and even for* f *as large as 0.43, it requires* r *at least* $77Meg$.

4 Block-Level Sampling

Record-level sampling is quite wasteful since scanning one tuple off the disk is not much faster than scanning the entire group of tuples that are stored in the same disk block. In this section, we consider the issue of exploiting all the tuples present in disk blocks that are scanned while performing the random sampling, without falling prey to the bias that can be introduced by correlations between the various values stored inside a disk block. We present an adaptive algorithm that adapts to the correlations in disk blocks and guarantees near-optimal sampling for ensuring specified bounds on the max error. We base this algorithm on the technique of cross-validation in statistics, combined with a generalization of Theorem 4 to the setting of cross-validation.

4.1 Challenges in Block-Level Sampling

The problem with using the entire block of tuples in the random sample is that it is possible that there are strong correlations between data values in the various tuples present in

a disk block. This could severely bias the approximate histogram being constructed. Consider the following different scenarios of correlation between tuples in a block: **a)** the tuples in each disk block are totally uncorrelated; **b)** the tuples in each disk block are totally correlated, e.g., have the same value in the attribute of interest, or the data is sorted on the attribute of interest and then assigned to disk blocks in that order; and, **c)** some fraction of the disk blocks exhibit correlated behavior.

Let the number of tuples in each disk block be b. In scenario (a), if we use all b tuples from a disk block when choosing a random sample, there is no loss in terms of the error in the histogram, and effectively we need to sample only r/b disk blocks to get the effect of sampling r tuples in record-level sampling. On the other hand, in scenario (b) suppose that each disk block contains tuples which all have the same value for the attribute of interest. Now, if we use all these tuples in the random sample, our effective sampling rate is one tuple per block; that is, we will need to scan as many as r disk blocks (a total of rb tuples) to get the same error in the histogram as we would obtain by using r tuples in record-level sampling. Finally, in scenario (c), suppose 20% of the disk blocks contain correlated tuples, while the remaining 80% contain a random collection of tuples. Now, we would expect that our effective sampling rate is close to 80%, i.e., we will have to sample $1.25 \times r/b$ disk blocks to get the effect of r sampling random tuples.

However, even if we were to generalize of our bound in Theorems 4 to block-level sampling, it is not clear how we can effectively perform the optimal amount of sampling for a given data distribution in disk blocks. The basic problem is that the distribution of data in disk blocks is typically not known a priori, and thus it is difficult to decide upon the right number of disk blocks to sample to guarantee the desired error bound for the approximate histogram.

4.2 Adaptive Sampling via Cross-Validation

A strategy for block-level sampling is to be adaptive by computing the number of random samples on-the-fly, with termination being based on some stopping rule which guarantees convergence to the desired error bound. Such strategies have been considered earlier in the literature: *adaptive sampling* by Lipton and Naughton [18], Lipton, Naughton, and Schneider [19], and Lipton, Naughton, Schneider, and Seshadri [20]; *double sampling* by Hou, Ozsoyoglu, and Dogdu [12]; and, *sequential sampling* by Haas and Swami [11]. A comparative evaluation of these methods can be found in Ling and Sun [17]. None of these methods is directly engineered for constructing histograms; they were designed at query cost estimation. All three methods compute a confidence interval on the error based on past sampling and terminate the sampling process when this interval is suitably small. They all make some assumptions about the data distribution or the sample size to obtain a confidence interval.

In contrast, we propose a novel adaptive sampling strategy based on the notion of *cross-validation* from statistics. This does not require any assumption about the data distribution. The idea is to sample groups of disk blocks iteratively, sampling g_i (specified later) blocks during the ith iteration. We maintain a histogram based on all tuples in all disk blocks sampled so far. During the ith iteration, after choosing g_i disk blocks at random, we determine the deviation error when partitioning the values in the g_i blocks with the separators of the current histogram. The process terminates if the deviation error is below a pre-computed threshold; otherwise, the histogram is updated to take into account the data in the most recently sampled blocks. In effect, we use the new sample to cross-validate the accuracy of the histogram, and if validation fails then we use the new sample to update the histogram. This update has the elegant feature that it has the most effect on the buckets with the largest error, thereby speeding up the rate of convergence. Another key feature of our adaptive algorithm is that it uses a random sample from the data to *test* convergence rather than making any a priori assumptions about the data distribution to decide when convergence can be *assumed*. In effect, our algorithm manages to use *the data distribution itself* (via the random sampling) to test for convergence on *that very data distribution*, without explicitly modeling the distribution. We summarize the algorithm below.

Adaptive Sampling Algorithm:

1. Compute r and $g_0 = \frac{r}{b}$ from n, f, k, γ, and b via Theorem 4.
2. Pick a random sample R consisting of g_0 disk blocks.
3. Using R, create an equi-height histogram H_0.
4. **repeat**
   (a) $i \leftarrow i + 1$.
   (b) Pick a new random sample R_i with $g_i = 2^{i-1} g_0$ disk blocks, and compute the max error δ_i in partitioning R_i using the separators of H_{i-1}.
   (c) Merge R_i with R. Build a histogram H_i from R.
5. **until** $\delta_i \leq f g_i / k$.
6. **output** current histogram H_i.

Our analysis suggest that a reasonable choice of the g_i values is $g_{i+1} = \sum_{j=0}^{i} g_j$, with $g_0 = g = r/b$; thus, $g_0 = g$, $g_1 = g$, $g_2 = 2g$, $g_3 = 4g$, ..., $g_i = 2^{i-1} g$. Suppose we can establish that sampling g_i blocks in the ith iteration is sufficient to determine the deviation error in the histogram at that stage *with respect to the actual data distribution*. (This is justified in Section 4.3 below.) We initially sample $g = r/b$ disk blocks; if the data is uniformly distributed without any correlation, this sample will suffice to guarantee the desired deviation error. The second stage using another $g = r/b$ random disk blocks will allow us to determine whether the deviation error is below the desired value, implying that we would have performed a near-optimal amount of sampling. In general, if the data distribution is such that we need to sample a total of xr/b disk blocks ($x \leq b$), then we achieve convergence at the iteration where g_i first exceeds xg, which again implies that we perform near-optimal amount of sampling. In any case, we are guaranteed to sample at most twice as many blocks as necessary for any given effective rate of sampling, in contrast to the naive strategy of sampling r disk blocks which is off by a factor b in the worse-case.

There are many twists on this basic strategy. For example, we could use only one randomly chosen tuple from each of the g disk blocks for purposes of cross-validation. It is also possible to be even more aggressive in adapting to the error in the histogram: we could adapt the choice of g at each stage of the iteration to speed up convergence without over-sampling by using larger values of g when the error is large relative to the total number of samples chosen up to

that point. A detailed exploration of these ideas, as well as a comprehensive analysis is deferred to the full paper [5].

4.3 Analysis of Adaptive Algorithm

The analysis of our adaptive algorithm needs some notation.

Definition 3 *Given a histogram H and a set $S \subset V$, let S_j be the number of values from S that lie in the jth bucket of H. Define the* relative deviation *of H with respect to S as*

$$\delta_S = \max_{1 \le j \le k} \left| |S_j| - \frac{|S|}{k} \right|.$$

That is, δ_S is the deviation we would get for H if we were to use the boundaries of H to define a histogram for S.

The following theorem, a generalization of Theorem 4, lays the theoretical foundation for our adaptive strategy. We defer the proof to the full paper [5].

Theorem 7

1. *Let H be a k-histogram for V with deviation $\delta = 2fn/k$, for $f \le 1/2$. Let S be a sample of size s from V, where*

$$s \ge \frac{4k \ln 1/\gamma}{f^2}.$$

   Then, the probability that $\delta_S \le fs/k$ is at most γ.

2. *Let H be a k-histogram for V with deviation $\delta = fn/2k$. Let S be a random sample of size s from V, where*

$$s \ge \frac{16k \ln k/\gamma}{f^2}.$$

   Then, the probability that $\delta_S \ge fs/k$ is at most γ.

Note that this theorem only considers deviation above the norm, but a similar theorem can be proved for the case of deviation below the norm [5]. Let us examine in detail the implications. The first part of the theorem says that given a histogram with max error $> 2fn/k$, a sample of size s is likely to have max error exceeding fs/k when it is partitioned according to this histogram. Conversely, the second part says that given a histogram with max error $\le fn/2k$, a random sample of size s is unlikely to have max error $> fs/k$ when partitioned according to this histogram. Thus, using Δ_S as a test, we can easily distinguish between the cases where the histogram has excessive error and negligible error. Thus, using a cross-validation based on S, we would never terminate the sampling too soon nor would be carry on sampling too long. The size of s needed to guarantee these properties is no larger than the sample size required to generate a histogram with max error $\le fn/k$.

Applying the argument to the iterations of the adaptive algorithm, with $S = R_i$ and H_{i-1} at stage i, we obtain a justification for our adaptive algorithm. We emphasize that this is merely a justification and not a proof, since at the initial stages we may not sample enough for cross-validation to work as desired. However, experiments (Section 7) indicate that this algorithm performs well in practice.

5 Handling Duplicate Values

In Section 2, we assumed that the set V does not contain duplicate values and our subsequent development was based on this assumption. However, in most applications it is the case that there are duplicates and sometimes of high multiplicity. In particular, when a value appears more often than n/k times, adjacent separator values may be same, i.e., $s_{j-1} = s_j = v$, implying that all items in B_j have the same value v. Moreover, there may be additional elements with the value v in preceding or succeeding buckets. While this may appear to be a minor technical point, it is really a tricky problem; for example, it is not even clear a priori how we should measure the error in the histogram or in fact what is the reference point for the error measurement. Specifically, since it is impossible to distinguish the duplicate values that appear in distinct bins, the problem of measuring b_j for the max error metric (Definition 1) is ill-defined. This problem directly impacts the cross-validation step of our adaptive sampling algorithm since we are no longer able to measure error for each bucket of the current histogram H_i.

One standard approach for dealing with this issue is to employ *compressed histograms*. Such histograms separate the representation of values of multiplicity higher than n/k from the rest of the values. We will consider this in detail in the full version [5] of the paper. Here we focus on a different approach based on generalizing the max error metric itself in the presence of duplicate values. Suppose that the separators obtained from the sample are $s_1, \ldots, s_{k-1}$. Let the sequence of *distinct* values in this sequence be $d_1, \ldots, d_m$, for $m < k$. Let the fraction of the sampled values that are less than or equal to d_j be

denoted by f_j, for $1 \le j \le m$. Similarly, let p_j be the fraction of data values that are less than or equal to d_j, for $1 \le j \le m$. We define the fractional max error f', generalizing f in Definition 1:

Definition 4 *Given a histogram with possibly duplicated separator values, the max error metric is defined to be: $f' = \max_{j=1}^{m} \frac{|(f_{j+1} - f_j) - (p_{j+1} - p_j)|}{f_{j+1} - f_j}$.*

In the context of adaptive sampling, the values f_j and p_j are obtained from the accumulated sample R and R_i respectively. When all values are distinct, $f_{j+1} - f_j = 1/k$ and $p_{j+1} - p_j$ reduces to b_j/n, and f' reduces to f as in Definition 1. However, when all values are not distinct, the factor $f_{j+1} - f_j$ measures the fraction of the data in the jth distinct range of the "current" histogram, whereas $p_{j+1} - p_j$ measures the observed fraction in the recently drawn sample. The denominator scales the error by the number of values in the jth distinct range. In the full paper, we discuss how the analytical results extend to this generalized error metric [5].

6 Estimating the Number of Distinct Values

Consider the problem of estimating the number of distinct values, say d, in V. Estimating the number of distinct values is an important subproblem in query optimization, e.g., for estimating projection size or in estimating relative error in join-selectivity estimation formulas used in System R [28]. We are interested in the issue of devising a good estimator for d based on a random sample from V.

In the statistics literature, the problem of estimating the number of distinct values (called the problem of estimating the number of species) has received a great deal of attention [1]. Recent work in the database literature includes the results of Naughton and Seshadri [22], and Haas, Naughton, Seshadri, and Stokes [10]. As observed by the latter, we obtain fairly poor performance when using the standard statistical estimators for the number of distinct values, e.g., the estimators due to Goodman [9], Chao [4], and Burnham and

Overton [2, 3] which have been used earlier in the database context in the work of Hou, Ozsoyoglu, and Taneja [13, 14] and Ozsoyoglu et al [25]. As Haas et al [10] remark, distinct value estimation is a hard and relatively unsolved problem, and few analytic results are available.

In this section, we first establish that no estimator for d can guarantee a small error. This explains the difficulty that past researchers have faced in trying to devise a good estimator for d using a random sample, so as to ensure a small error and low variance. Then, we propose a new estimator which we argue is optimal with respect to our negative result, and in practice is competitive with the ones studied earlier in the literature. We also propose a hybrid variant of our estimator which is expected to perform even better in practice. We also argue that the classical notion of error for an estimator of d is much stronger than necessary in database applications, and we propose a weaker notion which suffices for most practical purposes.

6.1 The Inherent Difficulty of Distinct Value Estimation

We use a variant of the standard notion of estimator error.

Definition 5 *The* error *of an estimator $\widehat{d}$ of d is defined as:*

$$error(\widehat{d}) = \begin{cases} \widehat{d}/d & \text{if } \widehat{d} \geq d \\ d/\widehat{d} & \text{if } \widehat{d} < d \end{cases}$$

Equivalently, the estimator $\widehat{d}$ has error at most α if and only if $d/\alpha \leq \widehat{d} \leq \alpha d$. The error is the *ratio* of the estimator and the number of distinct values d, where the ratio is inverted if necessary to ensure that the error is always more than 1.

As pointed out by Olken [23], all known estimators give exceedingly large errors on at least some of datasets. We show that large error is unavoidable even for relatively large samples *regardless of the estimator used.* That is, there does not exist an estimator which can guarantee reasonable error with any reasonable probability unless the sample size is very close to the size of the database itself. Another way of viewing the following result is that it implies that any estimator must necessarily have extremely high variance if it is to guarantee reasonable expected error.

Theorem 8 *Consider any estimator $\widehat{d}$ for the number of distinct values d based on a random sample of size r from a relation with n tuples. Then, for any $\gamma > e^{-r}$, there exists a choice of the relation such that with probability at least γ,*

$$error(\widehat{d}) \geq \sqrt{\frac{n \ln 1/\gamma}{r}}.$$

Consider the results of Haas et al [10] who obtained average error 1.33 and maximum error 2.86 (over a total of 24 high-skew distributions) using a random sample of size $r = 0.2n$. Setting $\gamma = 0.5$, we obtain that there exists a scenario where the error is at least 1.86. Our results are in fairly close accordance with real experiments both in terms of average error and the variance, even though we are giving worst-case error bounds and the experimental that may have had special structure that aids the estimator.

6.2 New Estimator and Error Metric

We complement the above negative result with a proposal for a new estimator which is simple and intuitive, and which can be argued to have near-optimal error. Suppose that we pick a random sample of size r from a relation with n tuples. Let f_j, for $1 \leq j \leq r$, denote the number of distinct values which occur exactly j times in the sample; clearly, $\sum_{j=1}^{r} f_j = r$. Our estimator is:

$$e = \sqrt{\frac{n}{r}} f_1^+ + \sum_{j=2}^{r} f_j,$$

where $f_1^+ = \max\{f_1, 1\}$. We give only a brief justification for this estimator, deferring a more detailed justification to the full paper [5]. Consider values whose frequency in the data is significantly more than n/r. We expect these values to occur more than once in the sample and therefore they are accounted for in the latter summation. The bad case involves values whose frequency in the data is anywhere between 1 and n/r. We expect each of these to occur at most once in the sample. Thus, each value contributing to f_1 "represents" a set of n/r values in which the number of distinct values could be anywhere between 1 and n/r. The first part of the estimator attempts to balance between the two extreme situations.

Theorem 8 limits our ability to determine the number of distinct values to any high degree of precision without scanning essentially the entire relation. On the other hand, estimators can be used with very high accuracy to determine whether d is significantly smaller than n or not. Such determination is useful in a number of optimization tasks, e.g., relative reduction in the size of a relation due to duplicate elimination. Formally, the error metric rel-error$(e) = \frac{d-e}{n}$ measures the estimation error for the number of distinct values d *relative to* n. In Section 7, we demonstrate that our proposed estimator e has a very low rel-error(e). The following numeric example illustrates the point. Let $n = 100,000$, $d = 500$, and $e = 5000$; now, the estimator e is off by a factor 10 with respect to d. However, using e to estimate d we would reach the correct conclusion that d is much smaller than n; observe that rel-error$(e) = 0.045$ indicating this fact. Thus, wherever the optimizer can exploit d/n instead of requiring an estimation of absolute d, it stands a better chance of realizing accurate estimation through sampling.

7 Experimental Evaluation

In this section, we present experimental results on Microsoft SQL Server 7.0. In the following, we refer to the algorithm presented in Section 4 as the CVB (for Cross-Validation based Block sampling) algorithm. Specifically, the results of this section demonstrate that:

a) Although the results in Section 3 assume true random sampling, the predicted relationship among number of histogram bins, sampling size and the error in approximation holds for block level sampling.

b) The CVB algorithm adjusts to varying data distributions gracefully. In particular, we studied:

- Practical convergence limits of the CVB algorithm for data with *varying skew*, from uniform distribution to highly skewed distribution.
- Effect of *partial clustering* of data on disk, i.e., the effect of clustering together a fraction of the tuples with the same data value.
- Effect of varying number of data records in a page.

c) The accuracy of our new estimator for number of distinct values as data distribution is varied (from "few" to "many" distinct values).

7.1 Implementation and Experimental Setup

The SQL Server optimizer uses equi-height histograms. As mentioned in Section 2, such histograms are characterized by their separators (step boundaries). In addition to histograms, the SQL Server also collects information on density [27]. However, because the estimation of the density was extremely accurate whenever the CVB algorithm converges, we defer a discussion of density estimation to the full version of the paper [5].

Implementation: The experiments were conducted on an Intel Pentium 200MHz processor with 64 MB RAM. The database was stored on external Seagate ST34371N disk drives. The CVB algorithm was implemented on Microsoft SQL Server 7.0. Unlike the earlier releases of SQL Server, SQL Server 7.0 makes it possible to exploit random sampling at the level of disk pages by specifying the percentage of file to be sampled. However, in order to implement CVB, we needed to modify the block-level server algorithm to make it adaptive. Two key extensions are:

1. For an incremental sample, sort the sample and compute the max error metric (Section 2). The algorithm terminates if this error is below a threshold.

2. Otherwise, the incremental sample is merged with the past samples using a merge algorithm.

While the theoretical results in the proposed algorithm recommends doubling of each successive incremental sample size from the point of view of an effective cross-validation, we experimented with a variety of stepping functions to tradeoff the efficiency of step (2) with the risk of oversampling. The details of experimentation with step sizes is deferred to the full version of the paper [5]. For the purposes of experiments in this section, the adaptive algorithm uses the following sizes for the successive random samples: $5isqrt(n)$, for $i = 1, 2, \ldots$, for accumulated samples. We also adapted the algorithm to produce approximate histograms when the accumulated sample size is $\sqrt(n)/i$, for $i = 5, \ldots, 1$.

Additional implementation for our experiments included: (1) Recording each step value of the histogram. (2) Recording, for each step value of the histogram, the number of rows in the sample that were less than or equal to the step value. (3) Counting and recording the number of distinct values in the sample. (4) Recording the density value, which is a measure of the average number of duplicates in the data. (Density 0.0 implies that all values in the column are distinct, while density 1.0 implies that all values in the column are identical). (5) SQL Server uses one disk page to store a histogram for a column. For an integer column this translates to 600 bins. We modified the histogram data structure so that the number of bins could be varied.

Data Generation: We generated data using the Zipf distributions [29]. The skewness parameter Z was varied. Although we studied several different values of Z varying from 0 to 4, we present results only for three values of Z — 0, 2, and 4. The distribution is uniform for $Z = 0$ and highly skewed for $Z = 4$, i.e., there are few values that occur very frequently. We varied the number of records in the table (N) from 5 million to 20 million. In all experiments but one (where we vary N), we present results for $N = 10^7$. The record sizes were varied from 16 bytes to 128 bytes. Such a variation made it possible to experiment with different blocking factors (i.e., number of records in a page). Finally, we experimented with two different layouts. In the first case, the data was clustered using tuple-ids which were generated at random. This allowed random placement on disk. However, in order to also consider layouts where layout is clustered or correlated, we generated a *partially clustered* data set as follows. Consider any Zipfian distribution. Assume that the distribution requires that for a value t, there should be n_t tuples. We created partially clustered data by generating for every distinct values $0.8 \times n_t$ of tuples with randomly generated distinct tuple-ids, but assigning the same tuple-id to $0.2 \times n_t$ of tuples. Subsequently, when we clustered the relation on tuple-id, we ensured that 20% of the duplicates were placed sequentially on the table. The full paper [5] will discuss extensive variations in layouts that were considered.

7.2 Experimental Results

Varying the number of records: In our first experiment we studied the effect of varying the number of records in the table on the required sampling rate. We fixed the max error to 0.1 and Z = 2, and varied the number of records between 5, 10, 15, and 20 million. Figure 3 shows that as the number of records in the table increases, a proportionately smaller percentage of rows need to be sampled to reduce the error below a given threshold. This behavior is consistent with our theoretical result in Section 3 which predicts that the required sampling rate drops at a rate of log(n)/n. Figure 4 shows that the number of *disk blocks* that need to be sampled to achieve a given error threshold remains almost constant as the number of records in the table is increased. This is also conforms to the expected behavior since Section 3 predicts that this dependence is proportional to log(n).

Effect of data distribution on error: In our second experiment, we studied the variation in the error metric as the sample size was increased for three different Zipfian distributions when the data layout was random. The number of bins was held constant at 600. We observe from Figure 5 that for all the three distributions, the convergence point is almost the same. Our result in Section 3 predicts that convergence is independent of the data distribution and the experimental results confirm this prediction.

Effect of bin size on sampling rate: Our third experiment studies the effect of bin size on the required sampling rate. For this experiment, we held the max error and the data distribution constant and varied the number of bins. Figure 6 shows the sampling rate required to achieve a max error of 0.2 as the number of bins is varied from 50 to 600. We observe from the graph that the required sampling rate increases in a linear fashion with the number of bins which matches our theoretical prediction.

Effect of clustering on sampling rate: In our next experiment, we kept the distribution constant and studied the variation in error metric with the sampling rate for the random and the partially-clustered layouts. As can be seen from Figure 7, when the data is partially sorted, a higher rate of sampling is required to achieve the same max error as in the random case. This indicates that the adaptive nature of the algorithm is able to correctly detect correlation and therefore samples more.

Effect of record size on sampling rate: We varied the number of records per block while fixing the number of records to one million. For a max error of 0.1, we found that as predicted, the required amount of sampling grows linearly with the record size. This can be seen in Figure 8.

Effectiveness in predicting distinct values: In our final experiment, we studied the effectiveness of our metric in predicting the number of distinct values in the data. We ran this experiment for two different data distributions, with N

fixed at 10 million. In the first distribution Z = 2, and the number of distinct values in the table were 6101. In the second experiment the data distribution was uniform with the additional constraint that each distinct value occurred 100 times. Thus the number of distinct values in the data was 100,000. We refer to this as the Unif/Dup distribution. Figures 9, and 10 compare the actual number of distinct values (numDVReal) with the number of distinct values in the sample (numDVSamp) and the number of distinct values estimated by our metric (numDVEst) for the two distributions respectively. Figures 11 and 12 show the variation in the *distinct value estimation error* with the sampling rate. We note that prediction is far more accurate for the Zipfian distribution compared to the Unif/Dup distribution since Zipf has fewer distinct values that are easily detected by a relatively small sample. However, in both cases, especially in case of the Unif/Dup distribution, the estimation error for the proposed metric is small.

References

[1] J. Bunge and M. Fitzpatrick. Estimating the Number of Species: A Review. *Journal of the American Statistical Association* 88(1993): 364–373.

[2] K.P. Burnham and W.S. Overton. Estimation of the size of a closed population when capture possibilities vary among animals. *Biometrika* 65(1978): 625–633.

[3] K.P. Burnham and W.S. Overton. Robust estimation of population size when capture possibilities vary among animals. *Ecology* 60(1979): 927–936.

[4] A. Chao. Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of Statistical Theory and Applications* 11(1984): 265–270.

[5] S. Chaudhuri, R. Motwani, and V. Narasayya. Using Random Sampling for Histogram Construction. Microsoft Research Report, In preparation, 1997.

[6] S. Chaudhuri and V. Narasayya. An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server. In *Proc. 23rd VLDB,* 1997.

[7] S. Finkelstein, M. Schkolnick, and P. Tiberio. Physical Database Design for Relational Databases. *ACM TODS*, 13(1988): 91–128.

[8] P.B. Gibbons, Y. Matias, and V. Poosala. Fast Incremental Maintenance of Approximate Histograms. In *Proc. 23rd VLDB,* pages 466–475, 1997.

[9] L.A. Goodman. On the estimation of the number of classes in a population. *Annals of Mathematical Statistics* 20(1949): 572–579.

[10] P.J. Haas, J.F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. 21st VLDB,* pages 311–322, 1995.

[11] P.J. Haas and A.N. Swami. Sequential Sampling Procedures for Query Size Estimation. In *Proc. ACM SIGMOD Conference,* pages 341–350, 1992.

[12] W. Hou, G. Ozsoyoglu, and E. Dogdu. Error-Constrained COUNT Query Evaluation in Relational Databases. In *Proc. ACM SIGMOD Conference,* pages 278–287, 1991.

[13] W. Hou, G. Ozsoyoglu, and B. Taneja. Statistical estimators for relational algebra expressions. In *Proc. 7th ACM Symposium on Principles of Database Systems,* pages 276–287, 1988.

[14] W. Hou, G. Ozsoyoglu, and B. Taneja. Processing aggregate relational queries with hard time constraints. In *Proc. ACM SIGMOD Conference,* pages 68-77, 1989.

[15] Y. Ioannidis and V. Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *Proc. ACM SIGMOD Conference,* pages 233–244, 1995.

[16] Y. Ioannidis and V. Poosala. Histogram-Based Solutions to Diverse Database Estimation Problems. *IEEE Data Engineering Bulletin* 18(1995): 10–18.

[17] Y. Ling and W. Sun. An Evaluation of Sampling-Based Size Estimation Methods for Selections in Database Systems. In *Proc. IEEE Conference on Data Engineering,* pages 532–539, 1995.

[18] R.J. Lipton and J.F. Naughton. Query Size Estimation by Adaptive Sampling. In *Proc. ACM PODS,* pages 40–46, 1990.

[19] R.J. Lipton, J.F. Naughton, and D.A. Schneider. Practical Selectivity Estimation through Adaptive Sampling. In *Proc. ACM SIGMOD Conference,* pages 1–11, 1990.

[20] R.J. Lipton, J.F. Naughton, D.A. Schneider, and S. Seshadri. Efficient Sampling Strategies for Relational Database Operations. *Theoretical Computer Science* 116(1993): 195–226.

[21] R. Motwani and P. Raghavan. **Randomized Algorithms.** Cambridge University Press, 1995.

[22] J.F. Naughton and S. Seshadri. On Estimating the Size of Projections. In *Proc. Third International Conference on Database Theory,* pages 499–513, 1990.

[23] F. Olken. *Random Sampling from Databases.* PhD Thesis, Computer Science, U.C. Berkeley, 1993.

[24] F. Olken and D. Rotem. Random Sampling from Databases – A Survey. Manuscript, 1995.

[25] G. Ozsoyoglu, K. Du, A. Tjahjana, W. Hou, and D.Y. Rowland. On estimating COUNT, SUM, and AVERAGE relational algebra queries. In *Proc. Conference on Database and Expert Systems Applications,* pages 406–412, 1991.

[26] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proc. ACM SIGMOD Conference,* pages 294–305, 1996.

[27] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proc. ACM SIGMOD Conference,* pages 256–276, 1984.

[28] P.G. Selinger, D.D. Astrahan, R.A. Chamberlain, R.A. Lorie, and T.G. Price. Access path selection in a relational database management system. In *Proc. ACM SIGMOD Conference,* pages 23–34, 1979.

[29] G.E. Zipf. **Human Behavior and the Principle of Least Effort.** Addison-Wesley Press, Inc, 1949.

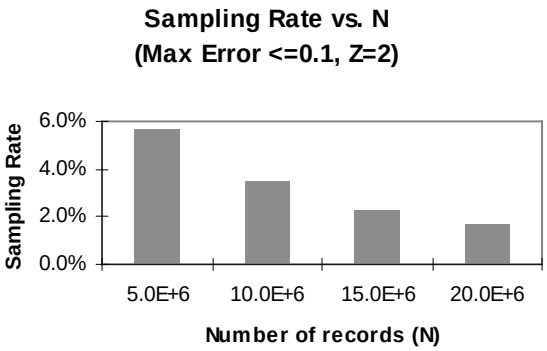

Figure 3. Variation of sampling rate with the number of records for a given error.

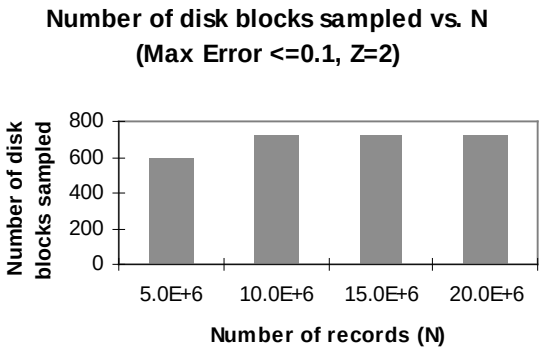

Figure 4. Variation of number of disk blocks to be sampled with the number of records for a given error.

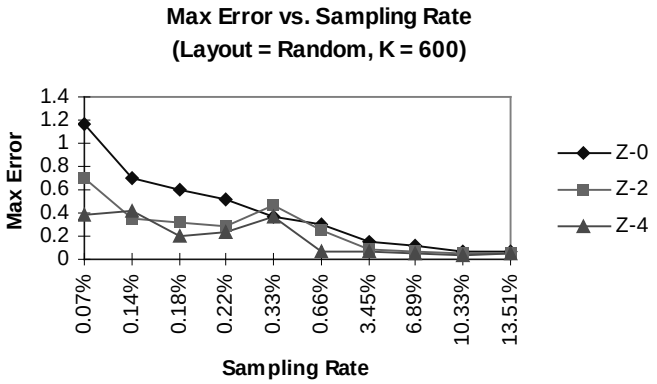

Figure 5. Effect of sampling rate on error for different data distributions.

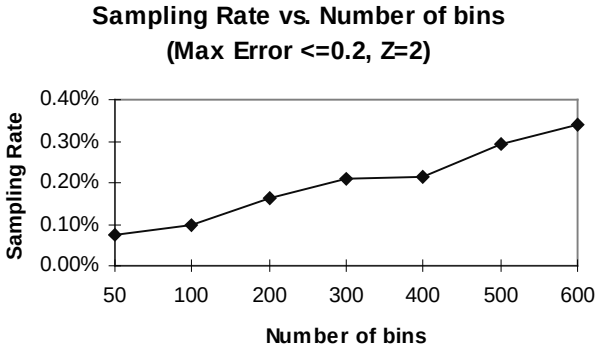

Figure 6. Effect of varying the number of bins on sampling rate for a given error.

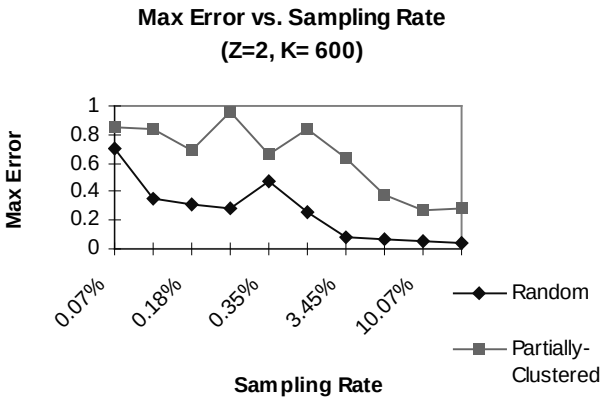

Figure 7. Variation in error vs. Sampling Rate for random and partially-clustered layouts.

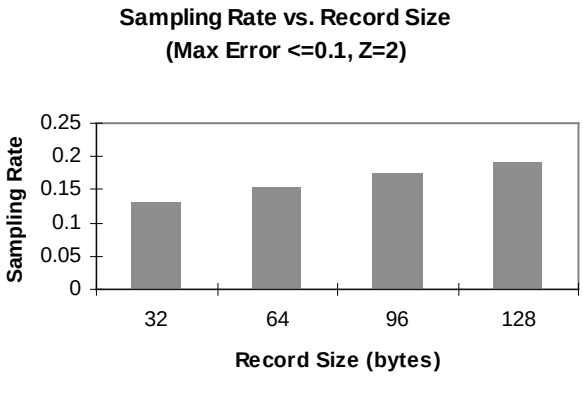

Figure 8. Variation in Sampling Rate with Record Size

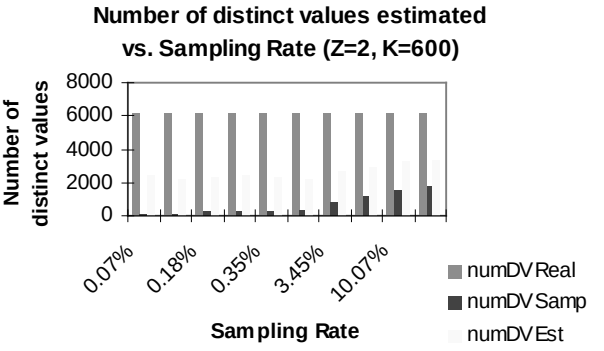

Figure 9. Variation in number of distinct values estimated with sampling rate (Z=2).

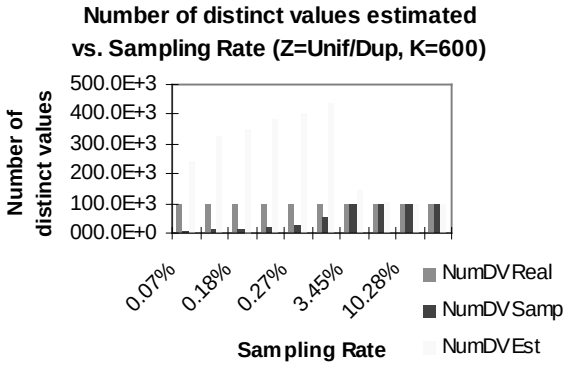

Figure 10. Variation in number of distinct values estimated with sampling rate. (Uniform distribution with duplicates)

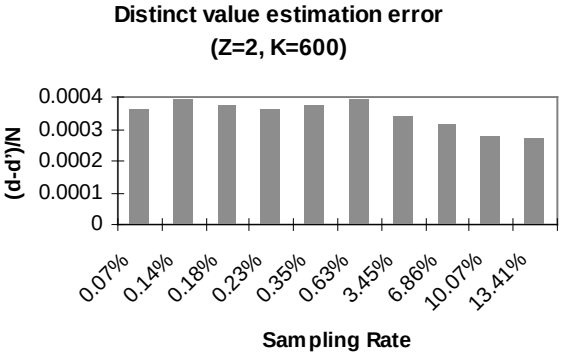

Figure 11. Difference between real and estimated number of distinct values vs. Sampling Rate.

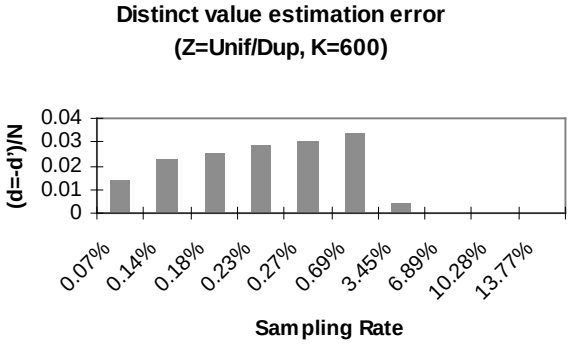

Figure 12. Difference between real and estimated number of distinct values vs. Sampling Rate.