Improved Histograms for Selectivity Estimation of Range Predicates

Viswanath Poosala
University of Wisconsin-Madison
poosala@cs.wisc.edu

Yannis E. Ioannidis*
University of Wisconsin-Madison
yannis@cs.wisc.edu

Peter J. Haas
IBM Almaden Research Center
peterh@almaden.ibm.com

Eugene J. Shekita
IBM Almaden Research Center
shekita@almaden.ibm.com

Abstract

Many commercial database systems maintain histograms to summarize the contents of relations and permit efficient estimation of query result sizes and access plan costs. Although several types of histograms have been proposed in the past, there has never been a systematic study of all histogram aspects, the available choices for each aspect, and the impact of such choices on histogram effectiveness. In this paper, we provide a taxonomy of histograms that captures all previously proposed histogram types and indicates many new possibilities. We introduce novel choices for several of the taxonomy dimensions, and derive new histogram types by combining choices in effective ways. We also show how sampling techniques can be used to reduce the cost of histogram construction. Finally, we present results from an empirical study of the proposed histogram types used in selectivity estimation of range predicates and identify the histogram types that have the best overall performance.

1 Introduction

Several modules of a database system require estimates of query result sizes. For example, query optimizers select the most efficient access plan for a query based on the estimated costs of competing plans. These costs are in turn based on estimates of intermediate result sizes. Sophisticated user interfaces also use estimates of result sizes as feedback to users before a query is actually executed. Such feedback helps to detect errors in queries or misconceptions about the database.

Query result sizes are usually estimated using a variety of statistics that are maintained for relations in the database. These statistics merely approximate the distribution of data values in attributes of the relations. Consequently, they represent an inaccurate picture of the actual contents of the database. The resulting size-estimation errors may undermine the validity of the optimizer's decisions or render the user interface application unreliable. Earlier work has shown that errors in query result size estimates may increase exponentially with the number of joins [IC91]. This result, in conjunction with the increasing complexity of queries, demonstrates the critical importance of accurate estimation.

Several techniques have been proposed in the literature to estimate query result sizes [MCS88], including histograms [Koo80], sampling [LNS90, HS95], and parametric techniques [CR94, SLRD93]. Of these, histograms approximate the frequency distribution of an attribute by grouping attribute values into "buckets" (subsets) and approximating true attribute values and their frequencies in the data based on summary statistics maintained in each bucket. The main advantages of histograms over other techniques are that they incur almost no run-time overhead, they do not require the data to fit a probability distribution or a polynomial and, for most real-world databases, there exist histograms that produce low-error estimates while occupying reasonably small space (of the order of 200 bytes in a catalog)[1]. Hence, they are the most commonly used form of statistics in practice (e.g., they are used in DB2, Informix, Ingres, Microsoft SQL Server, Sybase) and are the focus of this paper.

Although histograms are used in many systems, the histograms proposed in earlier works are not always effective or practical. For example, *equi-depth* histograms [Koo80, PSC84, MD88] work well for range queries only when the data distribution has low skew, while *serial* histograms [IC93, Ioa93, IP95] have only been proven optimal for equality joins and selections when a list of all the attribute values in each bucket is maintained. (In serial histograms, attribute values assigned to the same bucket need not be contiguous.)

In this paper, motivated by the above issues, we identify several key properties that characterize histograms and determine their effectiveness in query result size estimation. These properties are mutually orthogonal and form the basis for a general taxonomy of histograms. After placing all existing histogram types in the appropriate places in the taxonomy, we introduce novel techniques for several of the taxon-

*Partially supported by the National Science Foundation under Grants IRI-9113736 and IRI-9157368 (PYI Award), by Lockheed as part of an MDDS contract, and by grants from IBM, DEC, HP, AT&T, Informix, and Oracle.

[1] Nevertheless, one can construct data distributions that cannot be approximated well using a small number of buckets.

Abstract partitioning rule	Equi-width partitioning rule
• Adjoin to $\mathcal{T}$ a third column derived from the first two and sort $\mathcal{T}$ on it. Histogram buckets correspond to groups of elements of $\mathcal{T}$ that are contiguous in the order of the sorted third column.	• The third column is equal to the value column.
• Specify a restricted subclass of all possible histograms on a distribution $\mathcal{T}$, based on the number of elements of $\mathcal{T}$ allowed in each bucket, and consider only histograms in this subclass.	• There are no restrictions on the number of elements allowed in each bucket.
• Adjoin a fourth column derived from the first two.	• The fourth column is identical to the value column.
• Determine the unique partition of $\mathcal{T}$ into β buckets such that the histogram belongs to the restricted subclass and satisfies a specified constraint on the fourth column.	• Partition $\mathcal{T}$ so that the buckets contain attribute values in ranges of equal size.

Figure 1: Abstract partitioning rule and an example

omy dimensions, e.g., for assigning attribute values to buckets and approximating the data in a bucket, and then derive new histogram types by combining these techniques in effective ways. We also provide efficient sampling-based methods to construct several of the new histograms together with guidelines on the required sample size. Finally, we compare empirically the accuracy of both old and new histograms using a large set of data distributions and range queries. The results of these experiments identify the techniques that are most effective for each property in the histogram taxonomy, and point towards the histogram types with the best overall performance.

2 Histogram Definitions and Usage

The predicates that we consider are of the form $a \leq X \leq b$, where X is a non-negative real or integer-valued attribute in a relation R and a and b are constants such that $a \leq b$. Observe that such predicates include equality predicates (choose $a = b$) and "one-sided" predicates such as $X \leq b$ (choose $a = -1$).

2.1 Data Distributions

The *domain* $\mathcal{D}$ of X is the set of all possible values of X and the (finite) *value set* $\mathcal{V}$ ($\subseteq \mathcal{D}$) is the set of values of X that are actually present in R. Let $\mathcal{V} = \{\, v_i\colon 1 \leq i \leq D \,\}$, where $v_i < v_j$ when $i < j$. The *spread* s_i of v_i is defined as $s_i = v_{i+1} - v_i$, for $1 \leq i < D$. (We take $s_0 = v_1$ and $s_D = 1$.)

The *frequency* f_i of v_i is the number of tuples $t \in R$ with $t.X = v_i$. The *cumulative frequency* c_i of v_i is the number of tuples $t \in R$ with $t.X \leq v_i$, i.e., $c_i = \sum_{j=1}^{i} f_j$. The *data distribution* of X (in R) is the set of pairs $\mathcal{T} = \{\, (v_1, f_1), (v_2, f_2), \ldots, (v_D, f_D) \,\}$. Similarly, the *cumulative data distribution* of X is the set of pairs $\mathcal{T}^{\mathcal{C}} = \{\, (v_1, c_1), (v_2, c_2), \ldots, (v_D, c_D) \,\}$. Finally, the *extended cumulative data distribution* of X, denoted by $\mathcal{T}^{\mathcal{C}+}$, is the cumulative data distribution of $\mathcal{T}^{\mathcal{C}}$ extended over the entire domain $\mathcal{D}$ by assigning a zero frequency to every value in $\mathcal{D} - \mathcal{V}$.

2.2 Histogram Definition

A *histogram* on attribute X is constructed by partitioning the data distribution $\mathcal{T}$ into β (≥ 1) mutually disjoint subsets called *buckets* and approximating the frequencies and values in each bucket in some common fashion. The buckets are determined according to a *partitioning rule* that seeks to effectively approximate $\mathcal{T}$. (Note that this notion of a histogram is more general than the classical definition.)

In order to describe both new and existing partitioning rules in a uniform manner, we first present a multi-step abstract partitioning rule that captures the entire collection of partitioning rules in the paper (Figure 1). To illustrate our abstract definition, we also show how each step can be instantiated to yield the partitioning rule for classical *equi-width* histograms [Koo80]. In the description, $\mathcal{T}$ is viewed as a relation with two columns, the value column and the frequency column.

Based on the description in the table, every histogram is characterized by these properties:

1. *Partition class:* The restricted class of histograms considered by the partitioning rule.
2. *Partition constraint:* The mathematical constraint that uniquely identifies the histogram within its partition class.
3. *Sort parameter and source parameter:* The parameters derived from $\mathcal{T}$ and placed in its third and fourth column, respectively.

Each histogram is also characterized by the following additional properties:

4. *Approximation of values within a bucket:* The assumption that determines the approximate values within a bucket of the histogram.
5. *Approximation of frequencies within a bucket:* The assumption that determines the approximate frequency of each value within a bucket of the histogram.

Properties 4 and 5 determine the information that needs to be stored for each bucket. Note that all of the above properties are mutually orthogonal.

2.3 Histogram Maintenance and Usage

Typically, histograms are stored in system catalogs with the number of buckets limited only by the available disk space. Database updates are periodically propagated to histograms so that their effectiveness does not degrade. Techniques for determining appropriate schedules for such propagation are beyond the scope of this paper and do not affect the results presented here.

The *range* of a bucket $B \subseteq \mathcal{T}$ is the interval $[v_*(B), v^*(B)]$, where $v_*(B)$ and $v^*(B)$ are the smallest and largest attribute values covered by B. The *length* of its range is equal to $v^*(B) - v_*(B)$. To estimate the result size of the predicate $a \leq X \leq b$, an estimation routine identifies each bucket B for which the ranges $[v_*(B), v^*(B)]$ and $[a, b]$ overlap. Then, using specified approximation formulas, it estimates the number of values in each identified bucket that satisfy the range predicate, along with the frequency of each such value. These frequencies are summed over all identified buckets to yield the estimate of the result size.

3 Previous Approaches to Histograms

Several different histograms have been proposed in the literature. This section discusses the various choices that have been considered for instantiating the properties discussed above. The next section presents specific histograms as characterized by properties 1-3.

3.1 Partition Class

As indicated above, the histograms that we consider are *serial*[2] in the sense that histogram buckets correspond to groups of elements of $\mathcal{T}$ that are contiguous in the order of the sort parameter. Classical histograms (both "equi-height" and "equi-depth") have no constraints on the number of elements of $\mathcal{T}$ that can be assigned to a bucket. On the other hand, *end-biased* histograms [IC93, IP95] require that all but one of their buckets are singleton, i.e., they contain a single element of $\mathcal{T}$. One of the advantages of the end-biased histograms is their storage efficiency. As we will see later, singleton buckets occupy less space than buckets containing multiple attribute values. Hence, histograms with several singleton buckets (such as end-biased histograms) occupy less space than general serial histograms with the same number of buckets.

3.2 Partition Constraint

For the serial class, three different types of histograms have been defined, for various source parameters:

- *Equi-sum:* In an *equi-sum* histogram (with β buckets), the sum of the source values in each bucket is equal to $1/\beta$ times the sum of all the source values in the histogram.
- *V-optimal:* In a *v-optimal* histogram, a weighted variance of the source values is minimized. That is, the quantity $\sum_{j=1}^{\beta} n_j V_j$ is minimized, where n_j is the number of entries in the jth bucket and V_j is the variance of the source values in the jth bucket.
- *Spline-based:* In a *spline-based* histogram, the maximum absolute difference between a source value and the average of the source values in its bucket is minimized.

For the end-biased class, only the v-optimal histogram has been proposed, defined exactly as above.

3.3 Sort Parameter and Source Parameter

For the sort parameter, attribute values and frequencies have been proposed in the past. For the source parameter, spreads, frequencies, and cumulative frequencies have been proposed.

3.4 Approximation of Attribute Values and Frequencies

All histograms make the *uniform frequency* assumption and approximate all frequencies in a bucket by their average. Thus, all histograms require storage of the average frequency for each bucket.

Three different approaches exist for approximating the set of attribute values within a bucket. The most common is the *continuous values* assumption, where all possible values in $\mathcal{D}$ that lie in the range of the bucket are assumed to be present [SAC+79]. When $\mathcal{D}$ is an uncountably infinite set, (e.g., an interval of real numbers), the contribution of a bucket to a range query result size is estimated by linear interpolation. This assumption requires storage of the lowest and highest value in each bucket. Note that, for singleton buckets, this requires storing only one attribute value.

Another approach is the *point value* assumption [PSC84], where only one attribute value is assumed to be present (usually the lowest among those actually in the bucket). This assumption requires storage of this single attribute value. Finally, the histograms considered in [IP95] record every distinct attribute value that appears in each bucket (i.e., no assumptions are made). Such histograms require an auxiliary index for efficient access when estimating the result size of a query.

4 Previous Histograms

Several well-known and other relatively recent histograms are described in this section. Each one is primarily identified by its partition constraint and its sort and source parameters. If the choice in the above three properties is p, s, and u, respectively, then the histogram is named p(s,u). For s and u the abbreviations S, V, F, and C are used for spreads, attribute values, frequencies, and cumulative frequencies,

[2] Our current usage of the term is different from, and more general than, the usage in our earlier work [IC93, Ioa93, IP95]. In that work frequency was the only sort parameter considered.

respectively. When this definition is applied to a partition class other than *serial*, p is enhanced with the class name as well. Figure 2 provides an overview of the property combinations that have been proposed in the past.

SORT PARAMETER	*SOURCE PARAMETER*		
	SPREAD (S)	**FREQUENCY (F)**	**CUM. FREQ(C)**
VALUE(V)	EQUI–SUM	EQUI–SUM	SPLINE–BASED
FREQUENCY(F)		V–OPTIMAL	

Figure 2: Histogram Taxonomy.

Each of these histograms is discussed in a separate subsection. Within each bucket, each histogram makes the uniform distribution assumption for frequencies and usually the continuous values assumption for attribute values.

4.1 Trivial Histogram

Trivial histograms have a single bucket and vacuously belong to all histogram classes. They are equivalent to the popular *uniform distribution assumption*, used in most of the early and a few of the current database systems [SAC+79].

4.2 Equi-sum(V,S) alias Equi-width

Equi-sum(V,S) histograms group contiguous ranges of attribute values into buckets, and the sum of the spreads in each bucket (i.e., the maximum minus the minimum value in the bucket) is approximately equal to $1/\beta$ times the maximum minus the minimum value that appears in $\mathcal{V}$ [Koo80]. They are commonly known as *equi-width* histograms and are used in many commercial systems.

4.3 Equi-sum(V,F) alias Equi-depth

Equi-sum(V,F) histograms are like equi-width histograms but have the sum of the frequencies in each bucket be equal rather than the sum of the spreads [Koo80, PSC84]. They are popularly called *equi-depth* (or *equi-height*) histograms. If the frequency f_i of some value v_i is greater than the total frequency allowed for a bucket, v_i appears in multiple contiguous buckets, so that the total frequency of v_i (summed over all buckets in which v_i appears) equals f_i. Piatetsky-Shapiro and Connell [PSC84] considered equi-depth histograms in conjunction with the point value assumption and derived placements of the single point in each bucket for effective size estimation. Use of these histograms in commercial systems has been limited, because exact determination of "bucket boundaries" (i.e., the lowest and highest value in each bucket) can be very expensive. In Section 7, we discuss several approximate techniques for determining bucket boundaries that make practical implementation of essentially all types of histograms (including equi-depth histograms) feasible.

4.4 Spline-based(V,C)

Spline-based(V,C) histograms have not been actually proposed in the database literature, but are inspired by efforts in numerical analysis to approximate curves. Such a histogram is constructed effectively by obtaining a piece-wise linear approximation to $\mathcal{T}^{\mathcal{C}+}$. Since any range-query result size can be expressed in terms of cumulative frequencies, the better the approximation, the lower the result size estimation errors.

The problem of identifying optimal piecewise-linear approximations is known in numerical analysis as the *optimal knot placement* problem, which unfortunately, has no efficient solution [dB95]. We have adapted a heuristic algorithm due to deBoor [dB78]. Although rather complicated, the algorithm has very low time and space complexity; a detailed description appears elsewhere [dB78].

4.5 V-Optimal(F,F)

V-optimal(F,F) histograms group contiguous sets of frequencies into buckets so as to minimize the variance of the overall frequency approximation. In earlier work [IC93, Ioa93, IP95], they were simply called *v-optimal serial* histograms, and it was assumed that they would record every distinct attribute value that appeared in each bucket. The importance of these histograms is due to the fact that, under the above assumption and under a definition of optimality that captures the average over all possible queries and databases, these histograms have been proven to be optimal for estimating the result size of tree, function-free, equality join and selection queries [IP95]. The canonical construction algorithm involves an exhaustive (exponential-complexity) enumeration of all serial histograms and is clearly impractical. In Section 6, we show how to adapt a randomized algorithm to computing the v-optimal histogram.

4.6 V-Optimal-End-Biased(F,F)

V-optimal-end-biased(F,F) histograms are serial histograms in which some of the highest frequencies and some of the lowest frequencies are placed in individual buckets, while the remaining (middle) frequencies are all grouped in a single bucket. In earlier work [IP95], they were called *v-optimal end-biased* histograms. The importance of these histograms is due to their competitiveness with the v-optimal(F,F) histograms in many real-life situations [IP95]. The canonical construction algorithm involves an exhaustive enumeration of all end-biased histograms in slightly over linear time [IP95].

5 New Approaches to Histograms

None of the histograms described above are sufficiently accurate for general use in range query result size estimation. In this section, we propose several new choices for many of the histogram properties. We motivate each one by identifying the particular problem that it solves. The next section presents the specific combinations of these choices with which we experimented.

5.1 Partition Class

Biased histograms form an interesting class of histograms that falls between the serial and end-biased classes (i.e., it is a subclass of the former and a superclass of the latter). Biased histograms have at least one singleton bucket and possibly multiple non-singleton buckets. This new class allows systematic tradeoffs between the high accuracy of serial histograms and the low storage costs and computational efficiency of end-biased histograms.

5.2 Partition Constraint

We introduce two new partition constraints, which (as always) can be combined with various sort and source parameters. The goal of all the new partition constraints (and the earlier v-optimality constraint) is to avoid grouping vastly different source parameter values into a bucket.

- *Maxdiff:* In a *maxdiff* histogram, there is a bucket boundary between two source parameter values that are adjacent (in sort parameter order) if the difference between these values is one of the $\beta - 1$ largest such differences.
- *Compressed:* In a *compressed* histogram, the n highest source values are stored separately in n singleton buckets; the rest are partitioned as in an equi-sum histogram. In our implementation, we choose n to be the number of source values that (a) exceed the sum of all source values divided by the number of buckets and (b) can be accommodated in a histogram with β buckets. It turns out that most compressed histograms belong to the *biased* class.

5.3 Sort Parameter and Source Parameter

In most earlier serial histograms, the sort and source parameters have been either attribute values or frequencies, and the resulting histograms have been reasonably effective for approximating either value sets or frequency sets, respectively. The goal of any histogram, however, is to approximate well the entire data distribution $\mathcal{T}$, i.e., to approximate well both the value and frequency sets. Therefore, serial partitionings should contiguously group quantities that reflect proximity of both attribute values and frequencies. Toward this end, we introduce *area* as a possible choice for the sort and source parameters, defined as the product of the frequency and the spread. That is, the area a_i of v_i is given by $a_i = f_i s_i$. The area parameter is abbreviated below by A.

5.4 Approximation of Attribute Values Within a Bucket

One of the most serious drawbacks of previous histograms is their inaccuracy in approximating value sets with non-uniform spreads. As indicated by the experimental results in Section 8, the continuous values and point value assumptions used in previous histograms can lead to significant estimation errors.

To overcome this problem, we introduce the *uniform spread* assumption, under which each attribute value within a bucket is assumed to have a spread equal to the bucket average. This assumption requires storage of the lowest and highest value in each bucket together with the number of distinct attribute values in the bucket. The continuous values and point value assumptions also assume that attribute values have equal spreads. However, instead of storing the actual number of distinct values in each bucket, they make crude assumptions about it.

Example 5.1 Consider an equi-width histogram for an attribute with domain $\mathcal{D} = \{0, 1, 2, \ldots\}$. Assume that the range of a given bucket is equal to $[1, 100]$, the number of distinct values in it is equal to 10, and the sum of frequencies of attribute values in it is 200. Suppose we wish to estimate the result size for the range predicate $10 \le X \le 25$. Under the uniform spread assumption, the values in the bucket are $1, 12, 23, \ldots, 89, 100$, each having a frequency of 20, so that the estimated result size is 40. Under the continuous values assumption, the values in the bucket are $1, 2, \ldots, 100$, each value having a frequency of 2, so that the estimated result size is $16 \times 2 = 32$. Finally, under the point value assumption, the only value in the bucket is 1 (with a frequency of 200), so that the estimated result size is 0.

6 New Histograms

In this section, we introduce several new types of histograms obtained by specifying new choices for histogram properties as above or by combining earlier choices in novel ways. Figure 3 provides an overview of the new combinations that we introduce (enclosed in boxes) together with the earlier combinations discussed in Section 4. Note that all locations in the table correspond to valid histograms. We focus on histograms that intuitively appear to have good potential.

SORT PARAMETER	SOURCE PARAMETER: SPREAD (S)	FREQUENCY (F)	AREA (A)	CUM. FREQ (C)
VALUE (V)	EQUI–SUM	EQUI–SUM V–OPTIMAL MAX–DIFF COMPRESSED	V–OPTIMAL MAXDIFF COMPRESSED	SPLINE–BASED V–OPTIMAL
FREQUENCY (F)		V–OPTIMAL MAXDIFF		
AREA (A)			V–OPTIMAL MAXDIFF	

Figure 3: Augmented Histogram Taxonomy.

Each one of the new histograms is discussed in a separate subsection. All histograms make the uniform spread and the uniform frequency assumptions when approximating the data distribution within a bucket.

6.1 V-Optimal(V,F), V-Optimal(V,A), V-Optimal(A,A), and V-Optimal(V,C)

These histograms are identical to v-optimal(F,F) histograms, except that they use different sort and source parameters.

The v-optimal(V,F) and v-optimal(V,A) histograms minimize the variances in frequencies and areas respectively, while grouping contiguous attribute values. Using F (resp., A) as the source parameter ensures that skew in the frequency (resp., frequency and value) domains are considered in the bucketization, while using V as the sort parameter often results in a good approximation of the value domain.

By definition, the v-optimal(A,A) histograms minimize the variance of the overall approximation of area. Therefore, such a histogram should achieve a close approximation to both the value and frequency sets.

The reason for using the cumulative frequency parameter is somewhat different. Since the result sizes for any range query can be expressed in terms of cumulative frequencies, by grouping into buckets cumulative frequencies that are similar (v-optimal constraint), we should obtain a good approximation of $\mathcal{T}^{\mathcal{C}}$.

To avoid the exponential cost of the canonical algorithm to construct these histograms, we provide a randomized algorithm that with high probability finds a histogram close to the actual v-optimal histogram. The algorithm is applicable independent of the sort and source parameter choice. Experimentation with the Iterative-Improvement (II) and Two-Phase Optimization (2PO) randomized algorithms, which have been proposed as search strategies in query optimization [SG88, IK90], has shown that the simpler II algorithm produces very effective serial histograms, so we use II throughout. Details about II may be found in the above references. In our specific adaptation of II to the current problem, we define the *neighbors* of a histogram H to be all valid histograms that can be obtained by incrementing or decrementing a bucket boundary of H by one position in the domain of source values.

6.2 V-Optimal-End-Biased(A,A)

V-optimal-end-biased(A,A) histograms histograms are identical to the v-optimal-end-biased(F,F) histograms, except that they use area for the sort and source parameters.

6.3 Maxdiff(V,F), Maxdiff(V,A)

As mentioned in Section 5.2, the goal of all the new partition constraints is to avoid grouping attribute values with vastly different source parameter values into a bucket. The maxdiff histograms try to achieve this goal by inserting bucket boundaries between adjacent source values (in sort parameter order) that differ by large amounts.The motivations for using various sort and source parameters is exactly the same as those for the corresponding v-optimal histograms (Section 6.1).

These histograms can be efficiently constructed by first computing the differences between adjacent source parameters, and then placing the bucket boundaries where the $\beta - 1$ highest differences occur.

6.4 Compressed(V,F) and Compressed(V,A)

Compressed(V,F) histograms (resp., compressed(V,A) histograms) group contiguous attribute values into buckets, place the attribute values with the highest frequencies (resp., areas) in singleton buckets, and then divide the remaining values among multiple buckets in an equi-sum fashion.

By keeping values with high frequencies or areas in singleton buckets, these histograms achieve great accuracy in approximating the skewed frequency distributions and/or nonuniform spreads that are typical of many real-life data sets.

7 Some Computational Techniques

As can be seen from the above discussion, construction of the histograms considered in this paper requires, among other things,

- computation of "quantiles" (see definition below) for equi-depth histograms;
- computation of the frequency and cumulative frequency of each attribute value;
- computation of the number of distinct attribute values that lie in a given range; and
- computation of the spread of each attribute value.

In this section, we consider techniques for efficient computation of these quantities. We focus on methods that require at most one complete scan through the relation. (Such a scan is required when the data is initially loaded. Moreover, a complete scan is typically required by current DBMSs in order to compute quantities such as the largest and smallest key value in a column.) To be useful in practice, computational algorithms need to minimize the CPU cost per tuple of the relation, the number of I/O's required (over and above the complete scan), and the amount of main memory required for storage of intermediate results.

Throughout, we denote the number of tuples in the relation by N. We also denote by $f(v)$ the fraction of tuples in the relation with attribute value equal to v and by $F(v)$ the fraction of tuples with attribute value less than or equal to v. When v coincides with some $v_i \in \mathcal{V}$, we have $f(v_i) = f_i/N$ and $F(v_i) = c_i/N$.

7.1 Quantiles

To construct an equi-depth histogram with β buckets, we need to compute β bucket boundaries $q_1, q_2, \ldots, q_\beta$ such that $q_i = \min\{\, v \geq 0\colon F(v) \geq i/\beta \,\}$. In statistical terminology, q_i is the (i/β)-*quantile* of the attribute-value distribution. Exact computation of $q_1, q_2, \ldots, q_\beta$ requires sorting the entire relation in order of increasing attribute value. Then q_i is computed as the attribute value of the $r(i)$th tuple in the sorted relation, where $r = \lceil Ni/\beta \rceil$. Although this approach is simple and exact, it is too expensive to be used for the large relations typically encountered in practice. Internal sorting algorithms require too much main memory and CPU time, while external algorithms [GS91] incur multiple scans. We therefore focus on algorithms that compute approximate quantiles.

One well-known technique is the P^2 algorithm proposed by Jain and Chlamtac [JC85]. The basic idea behind this one-pass algorithm is to maintain a set of five "markers" that approximate the quantile of interest, the minimum value, the maximum value, and the two additional quantiles located midway between the quantile of interest and the maximum (resp., minimum) value. Whenever a new data value is read, the value of each marker is updated using a piecewise-parabolic curve-fitting technique. After the last data value is read, the middle marker is used as the estimate of the quantile. Raatikainen [Raa87] generalized the P^2 procedure to permit simultaneous estimation of more than one quantile during a single pass. As observed both in [Raa87] and in our own work, the accuracy of the procedure can be improved by using additional markers for each percentile to be estimated; we incorporate this enhancement into the version of the P^2 algorithm that we use in our experiments.

Another well-known approach to estimating quantiles is to use random sampling [MD88, PSC84]. The idea is to sample n ($\ll N$) tuples from the relation randomly and uniformly, without replacement. (Such a sample is called a *simple random sample*.) Then the quantile values for the sample are used as estimates of the corresponding quantile values for the entire relation. To obtain the random sample, we use a *reservoir sampling* algorithm due to Vitter [Vit85]. This algorithm (called Algorithm X in Vitter's paper) obtains a random sample of size n during a single pass through the relation. The number of tuples in the relation does not need to be known beforehand. The algorithm proceeds by inserting the first n tuples into a "reservoir." Then a random number of records are skipped, and the next tuple replaces a randomly selected tuple in the reservoir. Another random number of records are then skipped, and so forth, until the last record has been scanned. The distribution function of the length of each random skip depends explicitly on the number of tuples scanned so far, and is chosen such that each tuple in the relation is equally likely to be in the reservoir after the last tuple has been scanned. An advantage of the reservoir sampling approach is that it does not require the database system to support individual retrieval of randomly selected pages, and hence can be implemented in most current systems.

Both the sampling-based algorithm and the P^2 algorithm require exactly one pass through the relation, and hence have the same I/O cost. The intermediate storage requirements of these algorithms are also comparable. The P^2 algorithm, however, performs some fairly elaborate calculations for each tuple in the relation, while the sampling-based algorithm skips over most of the tuples in the relation, resulting in a low CPU cost per tuple. Unlike the P^2 algorithm, the sampling-based algorithm permits the subsequent adjustment of the histogram buckets required for constructing compressed histograms. Moreover only the sampling-based algorithm provides an estimate of the error in the approximation. Both of the above techniques can be extended to equi-sum histograms based on source parameters other than frequency.

7.2 Frequencies

Exact computation of frequencies and cumulative frequencies requires that a counter be maintained for each distinct attribute value and that each tuple in the relation be hashed on its attribute value and the appropriate counter be incremented. Such hashing can lead to excessive CPU costs. As with quantiles, the desired frequencies can be estimated from a random sample obtained using reservoir sampling. The estimated frequency of a value v_i is simply $n_i N/n$, where n_i is the number of tuples in the sample with attribute value v_i.

Of particular importance in histogram construction are the frequencies of the most frequent values. In some situations, it may be possible to obtain a very small "pilot" random sample of the tuples in the relation prior to the complete scan of the relation. Then, by adapting a technique due to Haas and Swami [HS95], the frequencies of the most frequent values can be obtained exactly with high probability. The idea is to obtain the pilot sample and observe the distinct attribute values that appear in the sample. During the full scan of the relation, the frequencies for these attribute values can be computed exactly using a relatively inexpensive hashing scheme. If the frequency of an attribute value is high, then with a very high probability the value will appear in the pilot sample, and the frequency of the value will be computed exactly. It is shown in [HS95], for example, that if the attribute values of a relation containing 10^6 tuples are distributed according to a Zipf distribution [Zip49] with parameter $z = 0.86$ (roughly, an "80-20" law), then with a probability of approximately 99.9% the 10 most frequent values will all appear in a sample of 1000 tuples (i.e., in a 0.1% sample). The more skewed the attribute-value distribution, the better the scheme works.

7.3 Distinct Values

Use of the uniform spread assumption (Section 5.4) requires techniques for computation of the number of distinct attribute values, denoted $d(l, u)$, that lie between given limits l and u. As with other statistics on the data distribution, exact computation of $d(l, u)$ typically requires too much CPU time and intermediate storage, due to the extensive hashing and/or sorting required.

The number of distinct values can be estimated based on a reservoir sample. The simplest procedure is to use the number of distinct values in the sample that lie between l and u, denoted $D(l, u)$, as an estimate of $d(l, u)$. Our experiments indicated that this simple estimate works reasonably well in practice. $D(l, u)$ typically underestimates $d(l, u)$ because some of the attribute values in the relation do not show up in the sample. The frequencies of the missing attribute values tend to be low, however, so that the absence of these values does not introduce serious errors into the final query-size estimate. In future work, we will investigate the utility of more sophisticated estimation methods such as those in [HNSS95].

7.4 Spreads

Histograms that use the area as a source and/or sort parameter require computation of the spreads of different attribute values. As with frequencies, spreads are expensive to calculate exactly. In our experiments, we simply used the spreads that appeared in a reservoir sample; this approach appeared to be adequate for our purposes. More sophisticated techniques for estimation of the spread can be derived directly from advanced techniques for estimation of the number of distinct values $d(l, u)$ (as defined in the previous section); this is a topic for future research.

7.5 Required Sample Size

The required sample size for various techniques presented in the previous sections depends upon the desired accuracy of the result size estimates. In general, there will be errors because the sample does not accurately represent the entire relation. For example, the relative error in estimating the result size for a predicate of the form $X \leq v$ is given by $|\widehat{F}_n(v) - F(v)|$, where $\widehat{F}_n(v)$ is the (random) fraction of tuples in a simple random sample of size n with attribute value less than or equal to v. We call such error the *sampling error*. In this subsection we discuss the relation between the sample size and the sampling error. To simplify the mathematics, we derive methods for determining the sample size under the (inaccurate) assumption that samples are obtained with replacement; for small sample sizes (<10%) this results in a slight overestimate of the number of samples needed.

A conservative estimate of the sample size required to control the relative sampling error to a desired level can be based on the following bound, originally due to Kolmogorov [Kol41]. Fix $n \geq 0$ and let $U_1, U_2, \ldots, U_n$ be a collection of independent and identically distributed random variables uniformly distributed on $[0, 1]$. For $0 \leq x \leq 1$, denote by $\widehat{F}_n^{(U)}(x)$ the (random) fraction of these random variables with values less than or equal to x. Finally, denote by G_n the distribution function of the random variable $\sup_{0 \leq x \leq 1} |\widehat{F}_n^{(U)}(x) - x|$. Then

$$P\left\{ \sup_{v \geq 0} |\widehat{F}_n(v) - F(v)| \leq \epsilon \right\} \geq G_n(\epsilon) \qquad (1)$$

for $\epsilon \geq 0$. Observe that G_n does not depend on either N, the size of the relation, or F, the form of the attribute value distribution function. The distribution G_n has been tabulated for small values of n by Massey [Mas51]; for large values of n (> 100), $G_n(x)$ is well-approximated by $G(n^{1/2}x)$, where $G(x) = 1 - 2\exp(-2x^2)$; cf [Mas51].

Consider a range predicate of the form $v_i \leq X \leq v_j$, where $v_i, v_j \in \mathcal{V}$ with $v_i \leq v_j$ (we allow equality predicates). The estimated result size based on a sample of size n is $N\left(\widehat{F}_n(v_j) - \widehat{F}_n(v_{i-1})\right)$. Thus, the relative sampling error R_n is

$$\begin{aligned} R_n &= \left(\widehat{F}_n(v_j) - \widehat{F}_n(v_{i-1})\right) - (F(v_j) - F(v_{i-1})) \\ &= \left(\widehat{F}_n(v_j) - F(v_j)\right) - \left(\widehat{F}_n(v_{i-1}) - F(v_{i-1})\right) \end{aligned}$$

Since, by the Triangle Inequality,

$$\begin{aligned} |R_n| &\leq |\widehat{F}_n(v_j) - F(v_j)| + |\widehat{F}_n(v_{i-1}) - F(v_{i-1})| \\ &\leq 2 \sup_{v \geq 0} |\widehat{F}_n(v) - F(v)| \end{aligned}$$

it follows from (1) that

$$\begin{aligned} P\{\, |R_n| \leq \epsilon \,\} &\geq P\{ \sup_{v \geq 0} |\widehat{F}_n(v) - F(v)| \leq \epsilon/2 \,\} \\ &\geq G_n(\epsilon/2). \end{aligned}$$

For example, a sample size of 1064 tuples is sufficient to give a relative sampling error of less that 10% with 99% probability. Similar arguments show that for one-sided predicates such as $X > v_i$, only about 270 samples are needed to achieve the above accuracy.

The above guidelines are conservative in that fewer samples are actually needed in practice to achieve a given degree of accuracy. (For example, Theorem 1 in Hoeffding [Hoe63] implies that only about 190 samples are actually needed to achieve the above accuracy for an equality predicate $X = v_i$ in the specific case $f(v_i) = 0.2$.) In our experiments, we used a sample size of 2000 tuples.

7.6 Construction Cost

Table 1 illustrates the difference in the construction costs of various histograms. It contains actual timings collected from running the corresponding algorithms on a SUN-SPARC, for varying amounts of space allocated to the histograms. The specific timings are for histograms with V and A as the sort and source parameters, respectively, but all other combinations of sort and source parameters produce quite similar results. These timings do not include the time taken to scan the relation and compute the sample. The cardinality of $\mathcal{T}$ (i.e., the number of distinct values) was fixed at 200 and the total number of tuples was equal to 100,000. All but the equidepth-P^2 histograms were constructed based on a reservoir sample of 2000 tuples. As can be seen from Table 1, the construction cost is negligible for most of the histograms when sampling techniques are used. The P^2 algorithm is expensive because of the significant processing cost incurred for each tuple in the relation.

Histogram	Time Taken (msec)	
	Space = $160b$	Space = $400b$
Compressed	5.9	9.3
Equi-sum	6.2	10.9
MaxDiff	7.0	12.8
V-optimal-end-biased	7.2	10.9
Spline-Based	20.3	41.7
V-optimal	42.9	67.0
Equi-Depth: by P^2	4992	10524

Table 1: Construction cost for various histograms

8 Experimental Results

We investigated the effectiveness of different histogram types for estimating range query result sizes. The average error due to a histogram was computed over a set of queries and expressed as a percentage of the relation size. That is, for a set

$\mathcal{Q}$ of N queries, the error E was computed as

$$E = \frac{100}{N} \sum_{q \in \mathcal{Q}} \frac{|S_q - S'_q|}{S_q},$$

where S_q and S'_q are the actual and the estimated size of the query result, respectively. The histogram types, data distributions, and queries considered in our experiments are described below. Observations on the sensitivity of our results to the allotted storage space, sample size, and number of distinct values in the data distribution are presented in Section 8.4.

Histograms: Experiments were conducted using all histogram types described in Figure 3. In general, the bucket of a histogram contained four floating numbers: the number of elements in the bucket, the lowest and highest attribute value in the bucket, and the average frequency in the bucket. In special cases, however, buckets could occupy less space. In particular, buckets of histograms with V as the sort parameter contained only three floating numbers: the lowest attribute value in a bucket was not explicitly stored but was implicitly assumed to be the successor (in the attribute's domain) of the highest value of the previous bucket. Also, singleton buckets contained only two floating numbers: the single attribute value in it and the corresponding frequency.

Since different histograms need to store different amounts of information per bucket, the number of buckets varied among histogram types in our experiments, with differences of up to 50%. To ensure fair comparisons, all histograms were constructed so that they occupied the same amount of space. The amount of available space was fixed at 160 bytes (approximately 10 buckets for a general serial histogram with a sort parameter other than V and 20 buckets for an end-biased histogram). All histograms were constructed based on samples of 2000 tuples, except for the trivial histogram, the equi-depth histogram in which bucket boundaries were computed exactly by sorting all of the tuples in the relation, denoted by equi-depth:precise, and the equi-depth histogram constructed from all the tuples in the relation using the P^2 algorithm, denoted equi-depth:P^2.

Data Distributions: Experiments were conducted using synthetic data distributions with $100K$ to $500K$ tuples, and number of attribute values (D) between 200 and 1000. In order to isolate the effects of different choices for the various orthogonal parameters of histograms, we experimented with several frequency and value sets. The frequency sets were generated independently of the value sets, and different types of correlation were induced between each frequency set and value set, thereby generating a large collection of data distributions. The choices for each set and for their correlation are given below:

- *Frequency Sets*: These were generated with frequencies following a Zipf distribution, with the z parameter varied between 0 (uniform) and 4 (highly skewed), which allowed experimentation with several degrees of skew.
- *Value Sets*: All attribute values were nonnegative integers, and spreads were generated according to one of five alternative distributions: *uniform* (equal spreads), *zipf_inc* (increasing spreads following a Zipf distribution), *zipf_dec* (decreasing spreads following a Zipf distribution), *cusp_min* (*zipf_inc* for the first $D/2$ elements followed by *zipf_dec*), *cusp_max* (*zipf_dec* for the first $D/2$ elements followed by *zipf_inc*), and *zipf_ran* (spreads following a Zipf distribution and randomly assigned to attribute values). Example value sets following the above spread distributions are plotted in Figure 4. The default z parameter for the Zipf distributions was 2.

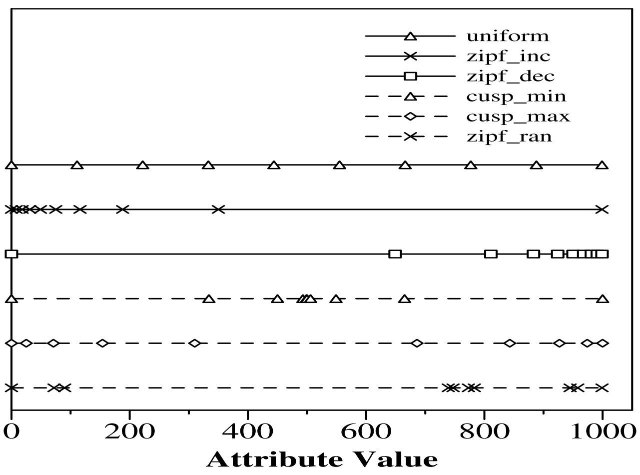

Figure 4: Value Sets.

- *Correlations*: Three different types of correlation were induced between the value and frequency sets. For *positive correlation*, values with high (resp., low) frequencies were mapped to values with high (resp., low) spreads. For *negative correlation*, high (resp., low) frequencies were mapped to values with low (resp., high) spreads. For *zero correlation*, frequencies were mapped to spreads randomly. In all cases involving random correlations, the average of errors in 10 runs of the experiment for different random mappings was used.

Queries: Experiments were conducted using five different query sets. All queries were of the form $a < X \leq b$; the sets differed in the values of the constants a and b. Set *A* contained all possible queries with $a = -1$ (so that the queries were one-sided) and b an integer lying between the minimum and maximum values in $\mathcal{V}$. (Observe that b assumed values in $\mathcal{V}$ and $\mathcal{D}$-$\mathcal{V}$.) Set *B* contained all queries with $a = -1$ and $b \in \mathcal{V}$. Set *C* contained 1000 "low-selectivity" random queries with $a, b \in \mathcal{V}$ and selectivities uniformly distributed in $[0, 0.2]$. Set *D* contained 1000 "high-selectivity" random queries with $a, b \in \mathcal{V}$ and selectivities uniformly distributed in $[0.8, 1]$. Set *E* contained 1000 "mixed-selectivity" random queries with $a, b \in \mathcal{V}$ and selectivities uniformly distributed in the composite interval $[0, 0.2] \cup [0.8, 1]$. The results of our experiments did not vary significantly for different query sets, so we only present those obtained for query set A below.

8.1 Typical Performance

It turned out that the relative performance of various histograms was fairly constant over a wide range of data and query sets. Due to the large number of combinations of the testbed parameter choices, we present results from one experiment that illustrates the typical behavior of the histogram errors. In this experiment, the value set follows *cusp_max*, the frequency set follows a Zipf distribution with parameter $z = 1$, and the correlation between values and frequencies is random. Table 2 shows (in decreasing order) the errors generated by the entire set of histograms on the query set A. As indicated by Table 2, a clear separation was observed

Histogram	Error (%)
Trivial	60.84
Equi-depth:P^2	17.87
V-optimal(A,A)	15.28
V-optimal(V,C)	14.62
Equi-width	14.01
V-optimal(F,F)	13.40
V-optimal-end-biased(A,A)	12.84
V-optimal-end-biased(F,F)	11.67
Equi-depth:Precise	10.92
Spline-based(V,C)	10.55
Compressed(V,A)	3.76
Compressed(V,F)	3.45
Maxdiff(V,F)	3.26
V-Optimal(V,F)	3.26
Maxdiff(V,A)	0.77
V-Optimal(V,A)	0.77

Table 2: Errors due to histograms

throughout the experiments between a set of effective histograms and a set of poor histograms. Although the relative performance of histograms in the lower set varies between experiments, and on some occasions histograms from the upper set were competitive, the results in Table 2 are quite characteristic overall. Hence, in the remaining subsections, we focus on the histograms in the lower part of the table. Also, the bottom part of the table demonstrates the importance of using V as the sort parameter.

The effectiveness of sampling is clearly illustrated by comparing the accuracy of the equi-depth:P^2 and compressed(V,F) histograms. As shown in Section 7.6, sampling-based construction of a compressed histogram requires much less CPU time than construction of an equi-depth:P^2 histogram. As can be seen from Tables 1 and 2, use of sampling and readjustment of bucket boundaries results in a histogram that is not only much cheaper to compute, but is far more accurate than the equi-depth:P^2 histogram. (Other experiments, not reported here, indicated that even ordinary equi-depth histograms computed from a sample are both cheaper to construct and more accurate than equi-depth:P^2 histograms.)

8.2 Effect of Frequency Skew

To study the effect of frequency skew, we present results from the experiments in which the value set has uniform spreads while the frequency set follows a Zipf distribution with varying skew (z). Note that the use of either frequency or area as the source parameter makes no difference here, since the value spreads are uniform. The histogram errors for queries in set A are plotted in Figure 5, with the value of the Zipf parameter z indicated on the x-axis and average error indicated on the y-axis. Clearly, for very low ($z = 0$) and very high ($z = 4$) skew, all histograms generate essentially no error. The hardest to deal with were intermediate skew levels. Since there were many frequencies that were quite different in these distributions, grouping them into only 10 or 15 buckets was insufficient to avoid errors. Overall, the maxdiff(V,A) and v-optimal(V,A) histograms performed better than the compressed(V,A) histograms.

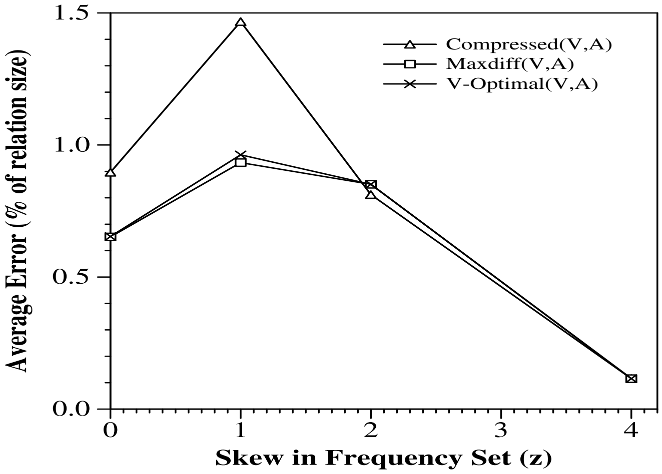

Figure 5: Frequency Set Approximation.

8.3 Effect of Non-uniform Spreads

To study the effect of non-uniform spreads, we present results from the experiments in which the frequency set is uniform while the value set follows *zipf_dec* with varying skew (z). The histogram errors for queries in set A are plotted in Figure 6, with the Zipf parameter z indicated on the x-axis. Clearly, there is a major advantage to using area as the source parameter, for all histogram types. Note that the v-Optimal(V,A) histogram performs consistently better than the others.

8.4 Sensitivity Analysis

In order to study the effect on histogram accuracy of the of the storage space, the sample size, and the number of attribute values in the relation, we conducted the above experiments for different values of these parameters (and their combinations). The results are shown for the v-optimal(V, A) histograms in Table 3 and Figure 7. Table 3 demonstrates that the accuracy of these histograms is not significantly affected by modest changes in sample size and number of attribute values. The effect of storage space is plotted in Figure 7, with the number of bytes on the x-axis and errors on the y-axis. It is clear that, beyond a small value of 160 bytes (10 buckets for general serial histograms with a sort parameter other than

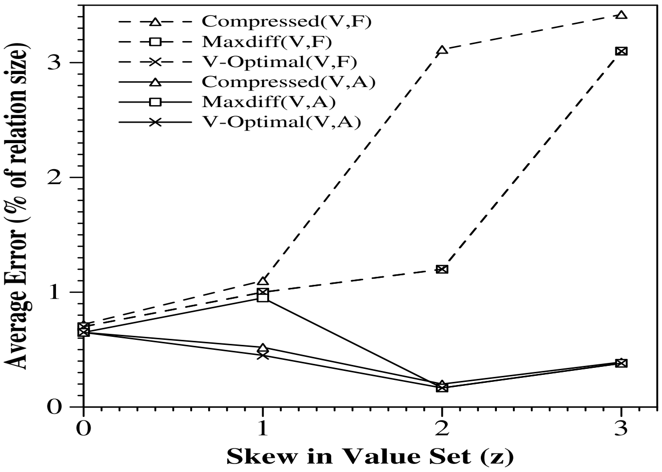

Figure 6: Value Set Approximation.

V), storage space has negligible effect on histogram accuracy. Also, it can be seen that the v-optimal and maxdiff histograms perform slightly better than the compressed histograms.

No. of Attr. Values	Error Sample Size 2000	10000	100000
200	0.59	0.46	0.29
500	0.68	0.49	0.31
1000	0.64	0.66	0.40

Table 3: Sensitivity Results

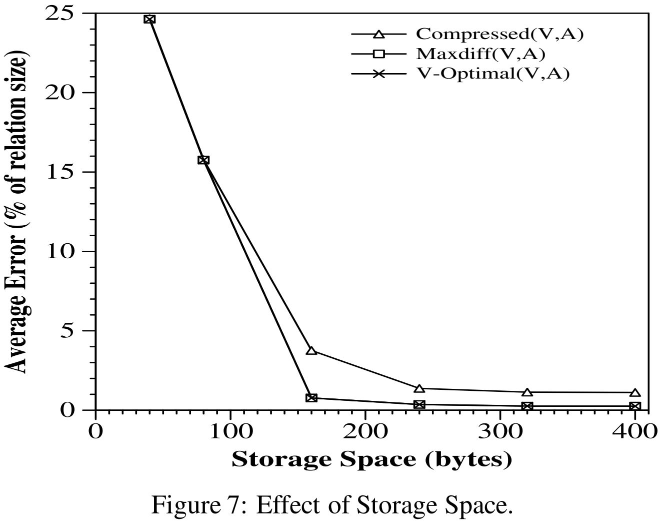

Figure 7: Effect of Storage Space.

8.5 Other Experiments

We observed in other experiments that all histograms perform slightly better when there is no correlation between spreads and frequencies, and that the uniform spread assumption approximates the value domain better than the continuous values and the point value assumptions.

9 Conclusions

This paper studied the use of histograms for estimating the result size of range queries. We systematically isolated and enumerated the various properties that characterize a histogram, some of which have never been considered explicitly in the literature. Motivated by our histogram taxonomy, we introduced the following innovations:

- explicit consideration of the spread between successive attribute values when assigning values to buckets;
- novel partition constraints that are more accurate than the traditional equi-sum constraint;
- use of the number of distinct values in a bucket to more accurately approximate the distribution of values and frequencies in the bucket;
- adaptation of a randomized algorithm for efficient construction of serial histograms; and
- use of reservoir sampling and statistical estimation techniques to efficiently construct histograms using a single scan of the data, together with guidelines on the required sample size that are tailored to range predicates.

Guided by our taxonomy, we combined both previous and new techniques in novel ways to yield several new histogram types. We then compared the accuracy of both old and new histograms empirically using a large set of data distributions and queries. The main conclusions from our experiments are as follows:

- The uniform spread assumption should be used to approximate the value set within each bucket.
- Area should be used as the source parameter.
- Attribute values should be used as the sort parameter.
- Equi-depth histograms constructed using reservoir sampling are both less expensive to obtain and more accurate than equi-depth histograms constructed using the P^2 algorithm. In general, sampling-based construction methods can produce accurate histograms at a small cost.
- The v-optimal(V,A), maxdiff(V,A), and compressed (V,A) histograms generate very small errors in a wide variety of situations. Over all data distributions, the v-optimal(V,A) and maxdiff(V,A) histograms performed better than the compressed(V,A) histograms. The computation times of the v-optimal histograms are slightly higher than those of the compressed and maxdiff histograms, but are still quite small (and quite insignificant compared to the costs of scanning the relation and generating the sample). Overall, we believe that maxdiff(V,A) is probably the histogram of choice, as it is very close to the best histogram on both issues, construction time and generated error.

References

[CR94] C. M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. *Proc. of ACM SIGMOD Conf*, pages 161–172, May 1994.

[dB78] C. de Boor. *A practical guide to splines*. Springer-Verlag, New York, 1978.

[dB95] C. de Boor. Private communication, 1995.

[GS91] A. P. Gurajada and J. Srivastava. Equidepth partitioning of a data set based on finding its medians. *Proc. Applied Computing Symposium*, pages 92–101, 1991.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. *Proc. of the 21st Int. Conf. on Very Large Databases*, pages 311–322, 1995.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

[HS95] P. J. Haas and A. N. Swami. Sampling-based selectivity estimation for joins using augmented frequent value statistics. *Proc. of IEEE Conf. on Data Engineering*, pages 522–531, 1995.

[IC91] Y. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. *Proc. of ACM SIGMOD Conf*, pages 268–277, 1991.

[IC93] Y. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM TODS*, 1993.

[IK90] Y. Ioannidis and Y. Kang. Randomized algorithms for optimizing large join queries. In *Proc. of the 1990 ACM-SIGMOD Conference on the Management of Data*, pages 312–321, Atlantic City, NJ, May 1990.

[Ioa93] Y. Ioannidis. Universality of serial histograms. *Proc. of the 19th Int. Conf. on Very Large Databases*, pages 256–267, December 1993.

[IP95] Y. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. *Proc. of ACM SIGMOD Conf*, pages 233–244, May 1995.

[JC85] R. Jain and I. Chlamtac. The P^2 algorithm for dynamic calculation of quantiles and histograms without sorting observations. *Communications of the ACM*, pages 1076–1085, Oct 1985.

[Kol41] A. N. Kolmogorov. Confidence limits for an unknown distribution function. *Ann. Math. Statist.*, 12:461–463, 1941.

[Koo80] R. P. Kooi. *The optimization of queries in relational databases*. PhD thesis, Case Western Reserver University, Sept 1980.

[LNS90] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. *Proc. of ACM SIGMOD Conf*, pages 1–11, May 1990.

[Mas51] F. J. Massey. The Kolmogorov-Smirnov test for goodness-of-fit. *J. Amer. Statist. Assoc.*, 46:68–78, 1951.

[MCS88] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):192–221, Sept 1988.

[MD88] M. Muralikrishna and David J Dewitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. *Proc. of ACM SIGMOD Conf*, pages 28–36, 1988.

[PSC84] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. *Proc. of ACM SIGMOD Conf*, pages 256–276, 1984.

[Raa87] K. E. E. Raatikainen. Simultaneous estimation of several percentiles. *Simulation*, 49:159–164, October 1987.

[SAC+79] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.T. Price. Access path selection in a relational database management system. *Proc. of ACM SIGMOD Conf*, pages 23–34, 1979.

[SG88] A. Swami and A. Gupta. Optimization of large join queries. In *Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data*, pages 8–17, Chicago, IL, June 1988.

[SLRD93] W. Sun, Y. Ling, N. Rishe, and Y. Deng. An instant and accurate size estimation method for joins and selections in a retrieval-intensive environment. *Proc. of ACM SIGMOD Conf*, pages 79–88, 1993.

[Vit85] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Software*, 11:37–57, 1985.

[Zip49] G. K. Zipf. *Human behaviour and the principle of least effort*. Addison-Wesley, Reading, MA, 1949.