Piecewise Linear Histograms for Selectivity Estimation

Xiaohui Yu and Ada Fu
Department of Computer Science and Engineering
Chinese University of Hong Kong

Abstract

Selectivity estimation of queries is of critical importance to query optimization. In order to get accurate estimations, database management systems must maintain statistics to capture the underlying data distribution. Histograms are extensively used in commercial database systems for this purpose. Most current histogram techniques make the assumption that all values in a single bucket appear with the same frequency, which rarely holds true in practice. Though wavelet-based histograms have recently appear as a strong alternative, several problems inherent with the wavelet techniques prevent it from practical use in database systems. In this paper, we propose a new type of histograms called piecewise linear histograms. Frequencies of attribute values in a bucket of the piecewise linear histogram is fit by a line using linear least squares regression, and the coefficients are stored as synopsis of the underlying data distribution. Moreover, since finding the best partition of the domain into buckets is a NP-hard problem, we propose a heuristic to efficiently determine the boundaries of buckets. Experimental results show that the piecewise linear histograms significantly outperform wavelet-based histograms.

1 Introduction

In relational database systems, users specify *what* information is needed, but do not describe *how* this information should be retrieved by the DBMS. While this simplifies the job of a user, it leaves to the DBMS the responsibility of identifying an efficient execution strategy (*execution plan*) among the different ways to execute a query, which differs in the indexes used, the order in which various operators are executed, and so on. Most non-trivial queries have a large number of plans, which are equivalent in their final output but vary vastly in their costs, e.g., the amount of time that they need to run. As a result, the DBMS needs to perform highly complex *optimizations* in picking a plan, which may not be practical for a human to perform. Hence most DBMSs have a component called the *query optimizer* to perform this operation.

Most commercial query optimizers belong to the class of *cost-based optimizers*. In principle, a cost-based query optimizer enumerates a set of equivalent plans for the submitted query (typically based on a dynamic programming strategy [SAC+79]), and selects the plan with the least cost (the *optimal* plan) for execution. The most common metric chosen for for cost is the time taken for completely executing a query. In general, accurately computing the cost of a plan will require actually running the plan. Clearly, it is highly impractical to follow this strategy for the large number of plans generated in a query optimizer. Hence, most DBMS's use some form of statistics on the underlying data, such as the number of tuples, the number of values, the distribution of values, the correlation between value sets, and the distribution of tuples among secondary storage units, in order to estimate the cost of a query plan approximately but efficiently. Since these statistics are used to provide approximate estimates query costs, the validity of the optimizer's decisions may be affected. This is an extremely important issue for query optimization because cost of plans can differ considerably and choosing a suboptimal plan can result in severe performance degradation.

One important aspect of cost estimation is selectivity estimation, which is concerned with estimating the percentage of tuples in the table that satisfy a given query. The selectivity of a query depends on the *data distribution* of the underlying data in the database. For this purpose, most DBMSs maintain or dynamically compute the data distribution of attributes approximately and use this information in computing the selectivity and then the costs of the queries. The earliest example of such statistics is the *uniformity assumption* made in System-R, where the distribution of an attribute is approximated by a uniform distribution – all attribute values between the lowest and highest values in the attribute are assumed to have the same frequency, which is equal to the average of the frequencies. Earlier work has shown that errors in query result size estimates based on the uniformity assumption may increase exponentially with the number of joins in the query [IC91]. As a result, the plan costs are estimated inaccurately and the optimizer will often end up picking a suboptimal plan for large queries. In fact, many commercial systems encounter this situation in real-life use [Loh94] and have started using more extensive statistical information about the underlying data. Several techniques have been proposed in the literature to estimate query result size [MCS88], including histograms [Koo80], sampling [LNS90, HS95], and parametric techniques [CR94, SLRD93]. Of these, histograms approximate the data 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 advantage 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-word databases, there exist histograms that produce low-error estimates while occupying reasonably small space (of the order of a few hundred bytes in a catalog). Hence, they are the most commonly used form of statistics in practice (e.g., they are used in DB2, Informix, Ingres, Oracle, Microsoft SQL Server, Sybase) and are the focus of this paper.

Various kinds of histograms have been proposed, e.g., equi-width histograms, equi-depth histograms, compressed histograms [PIHS96], and the recently proposed wavelet-based histograms. In this paper, based on an analysis of these previous histogram techniques, we identify their deficiencies and a new sort of histograms based on piecewise linear regression. Freqencies inside a bucket are regressed by a straight line, instead of being simply treated as taking equal values. We show by experiments that the proposed histograms outperform wavelet-based histograms in terms of accuracy. Meanwhile, the time complexity for both the construction and estimation phases remain low, which is a desirable property for database systems.

The remaining sections are organized as follows. In Section 2, we briefly review the previous histogram techniuques. Section 3 introduces the construction and usage of piecewise linear histograms. We compare the piecewise linear histograms with wavelet-based histograms by experiments and the results are presented in Section 4. Section 5 concludes this paper.

2 Related Work

Before reviewing the exisiting histogram techniques, we first introduce the set of definitions we will use in the following sections. These definitions are in line with [PIHS96]. The *domain* $\mathcal{D}$ of an attribute X in relation R 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. The cardinality of $\mathcal{V}$ is V. Let $\mathcal{V} = \{v_i : 1 \le i \le D\}$, 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 $1 \le i < D$. (We take $s_0 = v_1$ and $s_D = 1$.) In this paper we only consider numerical attributes. A commonly used technique for constructing histograms on non-numerical attributes (such as string fields, etc.) is to use a function that converts these data types into floating point numbers before constructing a histogram. 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 $t.X \le v_i$, 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 $\mathcal{T}^C = \{(v_1, c_1), (v_2, c_2), \ldots, (v_D, c_D)\}$. The *range* of a bucket $\mathcal{B} \subseteq \mathcal{T}$ is the interval $[v_*(B), v^*(B)]$, where $v_*(B)$ and $v^*(B)$ are the smallest and largest values covered by B. The *length* of its range is equal to $v^*(B) - v_*(B)$. The boundaries for the i-th bucket is b_i and b_{i+1}. To estimate the result size of the predicate $a \le X \le b$, 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.

Histograms are well-known statistical approximations to data distributions. A histogram on attribute X is constructed by partitioning the data distribution $\mathcal{T}$ into disjoint subsets called *buckets* and approximating the frequencies and values in each bucket in some common fashion. The boundaries of these buckets are determined according to a *partitioning rule* that seeks to effectively approximate $\mathcal{T}$.

Various types of histograms have been proposed and deployed in commercial systems. The most popular ones are the *equi-width* and *equi-depth* histograms [PSC84, mRL98]. Both of these histograms group contiguous ranges of attribute values into buckets and assume that all attribute values within the range corresponding to a given bucket occur with equal probability. They only differ in the partitioning rule employed, as follows. In an equi-width histogram, the widths of all buckets' ranges are the same; in an equi-depth (or *equi-height*) histogram, the total number of tuples having the attribute values associated with each bucket is the same.

Though widely employed, these histograms make the *uniform frequency* assumption and approximate all frequencies in a bucket by their average. However, in practice this assumption rarely holds true. Proximate attribute values may not appear with identical (or even almost identical) frequencies. In cases where frequencies in a bucket are volatile, using this uniform frequency assumption may result in highly inaccurate estimations.

The *V-Optimal* histograms proposed by Ioannidis *et al* partially addresses this problem. They group contiguous sets of frequencies into buckets so as to minimize the variance of the overall frequency approximation [Ioa93, IP95]. Under the assumption that every distinct attribute value that appears in each bucket, and under the 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 tree, equality join and selection queries. However, the construction of these histograms involves sorting the frequencies and an exhaustive (exponential complexity) enumeration of all serial histograms and is clearly impractical.

Recently, Matias et al [MVW98] proposed to using wavelet techniques to build histograms. The idea is to apply wavelet decomposition to the data distribution to obtain a compact data synopsis that comprises a selected small collection of wavelet coefficients. The experimental results presented in [MVW98] show that the wavelet-based histograms provide substantial improvements in accuracy over previous approaches. Nonetheless, such histograms do not explicitly minimize the fluctuation of freqencies inside buckets and thus cannot guarantee the supriority the resulting histograms. Furthermore, the fast wavelet transform algorithm requires the domain size to be power of 2, which is too strict to satisfy in real-life databases.

3 Piecewise Linear Histograms

Rather than roughly assuming each value appears with equal frequency inside a bucket, we now attempt to approximate the data distribution inside a bucket. Parametric methods can be used, i.e., to approximate the actual distribution with a known distribution function, such as normal, Pearson family, and Zipf distributions. However, this method suffers from the following problems:

1. In order to use this method, the "shape" of the actual distribution inside every bucket must be known beforehand to choose a suitable model function, which is almost impossible in real-life applications.

2. Since distributions usually vary from bucket to bucket, there is no single distribution which can fit the distributions in all buckets. Selecting a suitable distribution function for each bucket is non-trivial.

3. The number of distinct values in a bucket are usually small. Under such circumstances where the sample size is small, the conventional parameter estimation methods are no longer valid, because they are developed under the assumption of large sample size.

So, we should not use parametric methods to solve this problem. Non-parametric methods such as curve fitting is a better choice in this setting where there is little a priori knowledge.In theory we can approximate the data distribution in a bucket with a polynomial of any order. We choose to use the simplest one, i.e., linear least squares regression. The rationale here is that the fitting using higher order polynomials may incur more cost, and does not necessarily lead to better estimations due to the problem of *overfitting*.

3.1 Construction

Suppose we want to approximate the data distribution with m buckets. The construction of the piecewise linear histogram consists of two steps, which are described below.

- **Partition**

  In this step, $m-1$ cutting points are determined to partition the domain $\mathcal{D}$ into m buckets. Since the total number of possible partitions is in the exponential order of the domain size $(2_{|\mathcal{D}|})$, determining the best partition is NP-hard. This is problem is known as optimal knot placement in the statistics literature and there is no efficient solution yet. Instead of seeking the best partitioning, we propose a heuristic which can perform well in this setting. We define the *slope* l_i as

  $$l_i = \frac{f_{i+1} - f_i}{d_{i+1} - d_i} \tag{1}$$

  The heuristic is *putting the cutting points at where the slopes change most abruptly*. It ensures that the number of abrupt changes of slopes inside a bucket is as little as possible, thus the frequencies are more amiable to linear regression.

  For implementation of this heuristic, we can simply get the second order derivative of the frequency series, and put the cutting points at where the second order derivative is the largest.

- **Regression**

  After partitioning the domain, we perform linear regression on the frequencies inside each bucket and store the coefficients obtained.

  $$f_i = pd_i + q + \epsilon_i \tag{2}$$

  where ϵ_i is the residual.

  The estimation of p and q can be computed by

  $$\begin{aligned} \hat{p} &= \bar{f} - \hat{q}\bar{d} \\ \hat{q} &= \frac{\sum_i (d_i - \bar{d})(f_i - \bar{f})}{\sum_i (d_i - \bar{d})^2} \end{aligned} \tag{3}$$

Since linear least squares regression explictly minimize the mean square error of estimation inside a bucket, we can conclude that the regression coefficients obtained this way will incur least erros in the 2-norm sense in selectivity estimation.

Cleary, the time complexity for both the partition and regression steps are $O(n)$, where n is the domain size.

3.2 Usage

After obtaining the coeffients, we can now deal with both range and equality queries. For instance, given a equality query $X = a$, we first identify the bucket that a belongs to, and then use the coefficients $\{p, q\}$ corresponding to the bucket to calculate the estimated frequency of a according to the following formula:

$$\hat{f_a} = p * a + q \tag{4}$$

The calculation is very simple. And the process of identifying the appropriate buckets can be done in time $O(m)$, where m is the number of buckets.

4 Experiments

In the previous section, we presented a new type of histogram for selectivity estimation. In this section, we evaluate the accuracy of this method and compare it with wavelet-based histograms, which have been claimed to be the most accurate histograms.

We use synthetic data to test the accuracy of histograms. The frequency set is generated according to *Zipf* distribution. For a relation size N and domain size D, the frequencies generated by the Zipf distribution are

$$f_i = N \frac{1/i^z}{\sum_{i=1}^{D} 1/i^z} \quad \text{for all } 1 \leq i \leq D. \tag{5}$$

The skew of the Zipf distribution is a monotonically increasing function of the z parameter, starting from $z = 0$, which is the uniform distribution. In the experiments, we set $z = 1$, which is the canonical form of Zipf distribution.

It has been observed by several linguists and other researchers that subjects as diverse as income distributions and frequency distributions of words are very similar to Zipf distributions. A common claim in the database literature is that many attributes in real-life databases contain a few domain values with high frequencies and several with low frequencies [Chr84], and hence can be modeled well by Zipf distribution.

We tested the histograms using three different query sets:
A: $\{X \leq b | b \in \mathcal{D}\}$
B: $\{X \leq b | b \in \mathcal{V}\}$
C: $\{X = b | b \in \mathcal{D}\}$
D: $\{X = b | b \in \mathcal{V}\}$

Different histogram methods need to store different types of information. The wavelet-based histograms store two numbers per coefficient, while our piecewise linear histograms need to store three numbers per bucket: the lower boundary of the bucket and the two coeffcients obtained by regression.

For comparison, in our experiment, we use the same setting as that used in [MVW98]. The spreads of the value set follow the $cusp_max$ distribution with Zipf parameter $z = 1.0$. The frequency set is generated according to the Zipf distribution with parameter $z = 0.5$. The number of distinct attribute values is 500, and the number of tuples is set to 10^5. We also use the same amount of storage for different methods. The default storage space is 42 four-byte numbers (to be in line with [MVW98]).

Table 1-3 give the errors of two methods for query sets A, B, and C.

Error Norm	Piecewise Linear	Wavelets
$\|\|e^{\text{abs}}\|\|_1/T$	1.55%	0
$\|\|e^{\text{abs}}\|\|_2/T$	2.01%	0

In addtion to the above experiments, we also studies the effect of storage space for different methods. Figure ? plots the result of one set of our experiments from query set C. The setting of value set and frequency set is the same as in the above experiments.

5 Conclusion and Future Work

In previous sections, we have discussed various techniques for selectivity estimation. Based on a review of histograms currently in use, we identified that the uniform frequency assumption is a source of error in estimation. We deal with this problem by proposing a new type of histograms called piecewise linear histograms. Experimental results verifies it superiority over wavelet-based histogram techniques.

In future work, we will further the investigation by doing additional experiments using different settings. Also, we would like to compare the accuracy of the h istogram with other leading histograms. Finally, we believe that the piecewise linear histograms should form a class of histograms, not limited to the instance presented in this paper. Modification of the partitioning heuristic and the curve-fitting method will result in new histograms.

References

[Chr84] S. Christodoulakis. Implications of certain assumptions in database performance evaluation. *ACM TODS*, 9(2):163–186, 1984.

[CR94] Chung-Min Chen and Nick Roussopoulos. Adaptive selectivity estimation using query feedback. In Richard T. Snodgrass and Marianne Winslett, editors, *SIGMOD'94*, pages 161–172. ACM Press, 1994.

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

[IC91] Yannis E. Ioannidis and Stavros Christodoulakis. On the propagation of errors in the size of join results. In *SIGMOD'91*, pages 268–277. ACM Press, 1991.

[Ioa93] Yannis E. Ioannidis. Universality of serial histograms. In Rakesh Agrawal, Seán Baker, and David A. Bell, editors, *VLDB'93*, pages 256–267. Morgan Kaufmann, 1993.

[IP95] Yannis Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *SIGMOD'95*, pages 233–244, 1995.

[Koo80] R. P. Kooi. The optimization of queries in relational databases. Ph.d. dissertation, Case Western Reserve University, 1980.

[LNS90] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. In Hector Garcia-Molina and H. V. Jagadish, editors, *SIGMOD'90*, pages 1–11. ACM Press, 1990.

[Loh94] Guy M. Lohman. *Private Communication.* 1994.

[MCS88] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):192–211, 1988.

[mRL98] Surmeet Singh manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proceedings of SIGMOD'98*, pages 426–435. ACM Press, 1998.

[MVW98] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998*, pages 448–459. ACM Press, 1998.

[PIHS96] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. In H. V. Jagadish and Inderpal Singh Mumick, editors, *SIGMOD'96*, pages 294–305. ACM Press, 1996.

[PSC84] G. Piatetsky-Sharpiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proceedings of SIGMOD'84*, pages 256–276. ACM Press, 1984.

[SAC+79] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In Philip A. Bernstein, editor, *Proceedings of SIGMOD'79*, pages 23–34. ACM, 1979.

[SLRD93] Wei Sun, Yibei Ling, Naphtali Rishe, and Yi Deng. An instant and accurate estimation method for joins and selection in a retrieval-intensive environment. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD'93*, pages 79–88. ACM Press, 1993.