Quantifying Uncertainty in Multi-Dimensional Cardinality Estimations

Andranik Khachatryan
Karslruhe Institute of Technology
76131 Karlsruhe, Germany
andranik.khachatryan@kit.edu

Klemens Böhm
Karslruhe Institute of Technology
76131 Karlsruhe, Germany
klemens.boehm@kit.edu

ABSTRACT

We propose a method for predicting the cardinality distribution of a multi-dimensional query. Compared to conventional 'point-based' estimates, distribution-based estimates enable the query optimizer to predict the cost of a query plan more accurately, as we show experimentally. Our method is computationally efficient and works on top of a histogram already in place. It does not store any information additional to the histogram. Our experiments show that the quality of the predictions with the new method is high.

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Database Management—*Query processing; Relational databases*

General Terms

Theory

Keywords

Cardinality Estimation

1. INTRODUCTION

Accurate query result-size estimation is essential to query optimization. Using such estimates, the optimizer can compute the costs of different query plans and choose the better one (with lower cost). When doing this, optimizers typically use the assumption that the query cost is linear against the cardinality. This assumption rarely holds in practice, and can lead to inaccurate cost estimates and bad plan choices. A distribution-based estimate is a probability distribution over possible cardinality values. Previous work [5, 1] addresses this problem only partially, focusing on unidimensional predicates or on special classes of queries. [4, 3] discusses query optimization without the linear cost assumption (coined least-expected cost optimization) in general terms, but without focusing on how to derive cardinality distributions.

We propose a method, called the Sample-based method, which estimates cardinality distributions for multi-dimensional queries. The Sample-based method operates on top of a multi-dimensional histogram. We for our part have used the STHoles histogram [2, 6]. It has generic bucket layout which subsumes most other histograms. Our method uses past query execution results to come up with a cardinality distribution. Our experiments show that the Sample-based method offers better cost estimates than the estimates obtained using the STHoles histogram and the linear cost assumption.

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

2. DEFINITIONS AND NOTATION

We model cardinality estimates as random variables. Formally, the randomness is due to compression needed to obtain the histogram. Due to the compression we have imperfect information about the initial data set. A cardinality distribution instead of a point estimate is a way to model the uncertainty which comes from such imperfect information.

We write $card(q)$ to denote the cardinality of the query q. $F(\cdot)$ denotes the cumulative distribution function of a random variable.

A query plan π has an associated cost function $v_\pi(\cdot)$. For a query q, we define the cost of the plan π as

$$cost(\pi) = E[v_\pi(card(q))] \quad (1)$$

This is the expected cost according to the random variable $card(q)$. Histogram buckets and queries are hyper-rectangles in the attribute-value space. The volume of a hyper-rectangle is denoted by $vol(\cdot)$. The cardinality of a region divided by its volume is its selectivity – $sel(\cdot)$.

2.1 The STHoles Histogram

The STHoles histogram stores a tree of non-overlapping buckets. The goal is to obtain a partitioning of the data domain into regions with close to uniform density. The bounding box of a histogram bucket, denoted by $box(\cdot)$, is a hyper-rectangle in the attribute-value space. Each bucket stores the number of tuples in it, excluding child buckets. The volume $vol(\cdot)$ of a bucket is the volume of its rectangular bounding box, excluding the volume occupied by the child nodes.

The histogram estimates the cardinality of a query, using the uniformity assumption, i.e., it assumes that the tuples

are uniformly distributed inside a bucket.

$$n(q) = \sum_{b \in S} n(b) \cdot \frac{vol(b \cap q)}{vol(b)} \quad (2)$$

Example 1. In Figure 1 the histogram has three buckets: b_c is a child of b which is a child of b_{root}. The query q intersects with all three buckets. According to (2) the cardinality of query q is:

$$n(q) = n(b_c) + n(b) \cdot \frac{vol(q \cap b)}{vol(b)} + n(b_{root}) \cdot \frac{vol(b_{root} \cap q)}{vol(b_{root})}$$

Note that we can write $vol(b \cap q)$ because the region covered by b does not include b_c. The same holds true with $b_{root} \cap q$. □

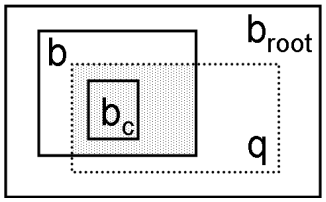

Figure 1: Query q and buckets b_{root}, b and b_c.

3. THE SAMPLE-BASED METHOD

The Sample-based method treats the past query execution results as a sample to approximate the random variable $card(q)$. Let X be a random variable; $\{x_1, \ldots, x_m\}$ is a sample for X. Then the cumulative distribution function of X can be approximated as follows:

$$F_m(z) = \frac{1}{m} \sum_{i=1}^{m} I(x_i \leq z) \quad (3)$$

where $I(P)$ is the "indicator" function, it equals 1 if the predicate P is true and 0 otherwise. [7] shows that F_m converges to F.

Example 2. Let m = 5, and $x_1 = 0.4$, $x_2 = 3$, $x_3 = 1.2$, $x_4 = 1.3$ and $x_5 = 1.9$. We now want to estimate the probability that $X \leq 1.2$. According to Equation (3),

$$F_5(1.2) = \frac{1}{5}(I(0.4 \leq 1.2) + I(3 \leq 1.2) + \\ + I(1.2 \leq 1.2) + I(1.3 \leq 1.2) + I(1.9 \leq 1.2)) = 2/5$$

□

Let b be the bucket which encloses q, and $b_1, \ldots, b_m$ are the child buckets of b. We use the already observed selectivities within the region of b to approximate the distribution of selectivities. These are:

- The child bucket selectivities: $sel(b_1), \ldots, sel(b_m)$.
- The selectivity which corresponds to the region covered by b, excluding child buckets:

$$s = \frac{n(b) - \sum n(b_i)}{vol(b) - \sum(vol(b_i))}$$

We approximate the cumulative distribution of selectivities inside the bucket using $\{s, sel(b_1), \ldots, sel(b_m)\}$ as a sample, using Formula (3). The buckets have different volumes, thus we weight the "evidence" with the relative volume of the bucket. For the selectivity s this is $(1 - \sum vol(b_i)/vol(b))$, for a child bucket b_i this is $vol(b_i)/vol(b)$. The formula for the cardinality distribution becomes:

$$Pr(sel(q) \leq x) = (1 - \sum_{i=1}^{m} \frac{vol(b_i)}{vol(b)}) \cdot I(s \leq x) \\ + \sum_{i=1}^{m} \frac{vol(b_i)}{vol(b)} \cdot I(sel(b_i) \leq x) \quad (4)$$

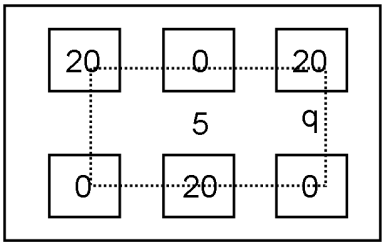

Figure 2: A histogram with query q

Example 3. Figure 2 shows a histogram which has 7 buckets (the root bucket has the largest bounding box) and a query which intersects with all buckets. Each of the child buckets spans ≈ 5% of the parent-bucket area. Three buckets have selectivity 0, three buckets have selectivity 20. The root bucket has selectivity 5. The probability that the selectivity is less than or equal to 7, according to (4), is:

$$Pr(sel(q) \leq 7) = 0.7 \cdot I(5 \leq 7) + 3 \cdot 0.05 \cdot I(0 \leq 7) + \\ + 3 \cdot 0.05 \cdot I(20 \leq 7) = 0.85 \quad \square$$

The Sample-based method uses only the information which is contained in the underlying histogram – it does not incur any additional storage costs. According to (4), the cumulative probability at any point can be computed at $O(m)$ cost.

4. EXPERIMENTAL EVALUATION

Conventional metrics used to evaluate point-estimation methods compare real and estimated cardinalities. This implicitly assumes a linear cost model. Instead, we adopt a metric based on the difference of the estimated and the real costs. To assess the quality of an estimator which issues cardinality distribution X, we compare the expected cost according to X to the real cost:

$$\epsilon_X = |1 - \frac{E[v_\pi(X)]}{v_\pi(c)}| \quad (5)$$

If we have two methods which yield distributions X and Y respectively, we compare ϵ_X and ϵ_Y. The smaller number indicates that the corresponding distribution is better.
We now describe an evaluation of the Sample-based method. We first describe our experimental setup in general terms; in Section 4.2 we describe the technical details; the experiments themselves are in Section 4.3.

4.1 Experiments – Overview

In the following, the cost of a physical operation is the number of I/O accesses performed. Because the metric in (5) depends on a cost function, we fix it to one that is both non-linear and common. Namely, the cost of a multi-pass hash-based join (HJ) of two relations, both having approximately size $M >> B$, is [8]:

$$cost(HJ) = O(M \cdot \log_B M) \quad (6)$$

For the evaluation we use a set of queries Q (more in Section 4.2) and calculate the normalized average error of the query-cost estimation for the Sample-based method and the STHoles:

$$\epsilon = \frac{1}{|Q|} \sum_{q \in Q} \epsilon_q \quad (7)$$

where ϵ_q is given by:

$$\epsilon_q = |1 - \frac{E[\pi(X)]}{\pi(c)}| \quad (8)$$

The method which produces the smaller normalized average error is better. For the distributions produced by the Sample-based method, the expected cost can be calculated using the following formula:

$$E[v(X)] = \sum_x v(x) \cdot Pr(X = x) \quad (9)$$

The original estimation method based on STHoles histograms issues point-based estimates. This means that the probability distribution consists of one value, $E[X]$, with probability 1. So, for STHoles $E[\pi(X)] = \pi(E[X])$, and we adjust (8) accordingly.

4.2 Experiments – Data

In our experiments, we vary the data sets, the query generation pattern, and the volume of the queries. Below, we describe these in detail. Overall our setup is similar to the one in [2].

We use three data sets, two of them are synthetic, one is based on the U.S. Census Bureau data set. The Census data set contains little over 210,000 tuples. Table 1 provides the parameters of the distributions used to obtain the synthetic data sets. The Array data set draws tuples from a Zipfian distribution. The data set contains 500,000 tuples, there are 50 distinct values per dimension, and the Zipfian skew equals 1. Because we use only 50 distinct values per dimension to generate 500,000 tuples, the Array data set has a lot of duplicate values.

The Gauss data set is a highly correlated, multi-dimensional real-valued data set. It consists of multi-dimensional, overlapping Gaussian bells with varying number of tuples in each bell. The queries in our workload span a certain percentage of the overall data domain. We vary the distribution of query centers. We write $Uniform$ to denote uniformly distributed query centers and $Data$ for a distribution that follows the data distribution. $Data$ means that we sample the data set to obtain the query centers. So $Gauss[Uniform, 1\%]$ means we are using the Gauss data set, and we generate query centers which are distributed uniformly over the data domain, each query spanning 1% of the overall data space. One run of our simulations consists of 1000 queries. We vary the maximal number of buckets in the histogram from 100 to 300. We plot the ϵ-metric for the Sample-based method and STHoles. The X-axis is the maximal number of histogram buckets, the Y-axis is the value of the ϵ-metric.

Data Set	Attribute	Value
Common	d: dimensionality	2
	N: cardinality	500,000
	data domain	$[0, \ldots, 1000]^d$
Array	distinct attribute values	50
	z:skew	1
Gauss	number of peaks	20
	standard deviation	50

Table 1: Description of data sets

4.3 Experiments

Figures 3, 4, 5, 6 show the *epsilon*-measures for different settings. Throughout the experiments, the Sample-based method performed better than the STHoles. In Figures 3 and 4 it has been a clear winner over the baseline point-estimation approach (the original STHoles estimation method). Figure 6 is the only setting where the refernce approach has slightly outperformed the Sample-based method. In all plots there is a slight slope noticeable – ϵ-measures start higher for 100 histogram buckets and decrease as the number of the buckets increases. This effect is expected, as the histograms usually fire better with more buckets. Summing up, the evaluation of the Sample-based method against a point-estimation method shows a significant improvement.

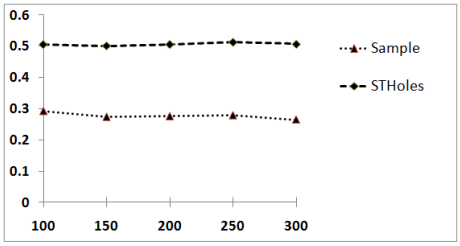

Figure 3: ϵ-measures for the $Gauss[Uniform, 1\%]$ setting

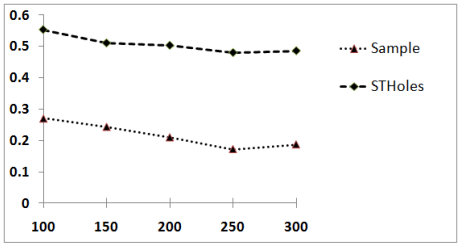

Figure 4: ϵ-measures for the $Array[Uniform, 1\%]$ setting

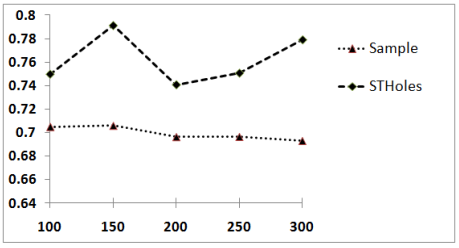

Figure 5: ϵ-measures for the $Census[Data, 1\%]$ setting

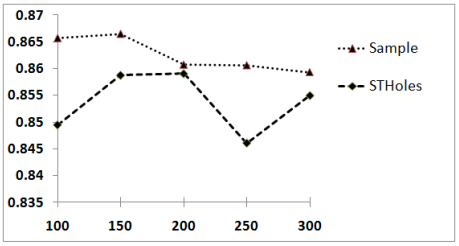

Figure 6: ϵ-measures for the $Census[Uniform, 0.5\%]$ setting

5. REFERENCES

[1] Babcock, B., and Chaudhuri, S. Towards a robust query optimizer: a principled and practical approach. In *SIGMOD '05*.

[2] Bruno, N., Chaudhuri, S., and Gravano, L. STHoles: a multidimensional workload-aware histogram. *SIGMOD Record, 2001*.

[3] Chu, F., Halpern, J., and Gehrke, J. Least expected cost query optimization: what can we expect? In *PODS '02*.

[4] Chu, F., Halpern, J. Y., and Seshadri, P. Least expected cost query optimization: An exercise in utility. In *PODS '99*.

[5] Donjerkovic, D., and Ramakrishnan, R. Probabilistic optimization of top n queries. In *VLDB '99*.

[6] Fuchs, D., He, Z., and Lee, B. S. Compressed histograms with arbitrary bucket layouts for selectivity estimation. *Inf. Sci. 177* (2007), 680–702.

[7] Kolmogorov, A. Confidence limits for an unknown distribution function. *Annals of Mathematical Statistics Volume 12, Number 4 (1941)* (1941), 461–463.

[8] L. M. Haas et al. Seeking the truth about ad hoc join costs. *VLDB Journal, 1997*.