Baku State University Journal of Mathematics & Computer Sciences ***2025****, v. 2 (3), p. 42-58*

journal homepage: http://bsuj.bsu.edu.az/en

IMPROVING MULTIDIMENSIONAL RANGE SELECTIVITY ESTIMATION USING ONE-DIMENSIONAL HISTOGRAMS

Chinar A. Aliyev[*]

Baku State University

Received 17 July 2025; accepted 25 August 2025

https://doi.org/10.30546/209501.102.2025.2.3.562

Abstract

Selectivity estimation is a critical task for query optimizer. Producing an efficient execution plan depends on that estimation. To estimate the result size of the query that includes multiple attributes needs to approximate of the joint data distribution (JD) of the attributes. in addition, it is not enough to approximate this JD, but also it could be used effectively to estimate selectivity of the range predicates of the attributes. Most commercial DBMS use attribute value independence assumption (AVI) to calculate selectivity. But in real world such assumption almost is not true and causes large errors during cardinality estimation process. In this paper we propose two histogram-based approach to estimate multi-attribute range selectivity. In both techniques one-dimensional histograms are used, however, joint frequency matrix (JFM) is also constructed and used when it is allowed to improve accuracy of estimation. Simplicity and accuracy allow these techniques to be implemented easily and practically more useful.

Keywords: nonlinear problem, eigenvalue problem, nodal solution, bifurcation point, global bifurcation
Mathematics Subject Classification *(2020): 34A30, 34B09, 34B15, 34C23, , 47J10, 47J15*

[*] Corresponding author.
E-mail address: z_aliyev@mail.ru (Ziyatkhan Aliyev)

1. Introduction

Accurately approximating the joint data distribution of multi-dimensional data set is a fundamental problem in query optimization. Histograms are widely used to approximate the data distribution in databases[1,2]. One-dimensional histograms have been studied very well and implemented in commercial database like Oracle, MS SQL, Postgres SQL and so on. But to approximate of multi-dimensional data and selectivity estimation using that approximation is not an easy task. The most commercial RDBMS use one-dimensional histograms to approximate JD and multiattribute selectivity estimation and when attributes are correlated then it can lead to huge errors and produce inefficient plan [11].

The first solution to solve the problem was multidimensional histogram and purposed by Muralikrishna M, DeWitt [3] as a multi-dimensional Equi-depth histogram(ED-MDH). Multidimensional histograms partition the data space into overlapping or nonoverlapping buckets and in each bucket, the data distribution is considered as uniform. ED-MDH partitions data space into buckets that each has approximately same number of tuples. It requires multiple scans over the relation to sort and construct the histogram. Although, the total cost is not linearly based on dimensionality but for large data sets it brings time and space complexity as well. In addition, to create the primary partitions, selecting of the first attribute from attribute set is heuristic.

It was improved by Poosala and Ioannidis[4] as PHASED and MHIST-2. The sconed is an adaptive version of the first one. It was suggested to select attributes based on *most in need of partitioning*. For V-Optimal histograms , it means a marginal distribution that has maximum variance of source parameter values. Although, this improves overall selectivity estimation but, still struggle with skewed data of JD due to the decision is based on marginal distribution.

V-Optimal histograms[13,14] have been well studied as well for one dimensional case, however multidimensional version of it is known to be NP-hard[15] and reducing density variance within buckets is expensive.

MinSkew[5] is a multidimensional histogram for spatial datasets and approximates the original dataset using regular grid and includes the basic idea of optimal histograms, avoids excessive constructing cost. The technique to reduce the complexity of the optimal partitioning problem, a special type of partitioning, binary space partitioning(BSP). Even being use BSP the complexity of the optimal histogram increases with the dimensionality. For this reasons, Min-Skew tries to find the best dimension for each bucket based on the marginal histograms. The

variance in each bucket is normalized with number of points in each bucket and it is minimized. The technique was proposed for two-dimensional set of rectangles but can easily be extended to higher dimensionals as well.

GenHist[6] allows buckets to overlap in the space of multidimensional data. It starts from a uniform grid of partitioning of the space and then periodically expands the buckets that has high number of elements. As a result, density of each bucket decreases and produces more smoother total density. Overlapping buckets allows to approximate the data distribution by using the combination of all buckets that related to the area. At the end, it generates low errors, however its construction time is slow.

Digithist[9] is a novel histogram-based structure for selectivity estimation of multidimensional range queries and **DigitHist** offers **theoretical guarantees** in the form of ε-approximations with respect to range queries. This is a strong **probabilistic quality assurance** on the accuracy of query answers, something many other histograms (like MinSkew or MHIST) do **not** explicitly provide.

There are also non-histogram-based approaches have been proposed [7,8] based on transformation and lossy compression techniques. Another type of non-histogram approach is random sampling [16]. To estimate the selectivity, a random subset of rows are read which can be done offline or online, but it can be skew sensitive, and its accuracy might depend on the increasing number of dimensionalities.

Multi-dimensional histogram-based estimation brings some problems like optimal partitioning which is NP-complete[12], optimal sort parameter which is suggested Hilbert numbering[4] but it also can cause large errors as well. The selectivity estimation for large data sets with high-dimensionals is still an open problem, because of above reasons and including large variance of selectivity, curse of dimensionality, construction complexity and so on.

Motivated by above problems we have investigated two ways to approximate JD using one-dimensional histograms and JFM when it is possible in straightforward way. The first one we called PHIST(partitioned histogram) is described below which allow us to understand how the attribute values are distributed over the JD. And the second approach is an improvement existing AVI assumption(RHIST – Related Histogram) by supplying additional information to support and improve accuracy of the multi-dimensional range selectivity estimation. The key ideas in this paper are based on extracting and producing necessary data and statistics from the grouping attribute value's combinations. This combination can be bult

via concatation of attributes or using hash function to produce unique values through one pass scan of the relation and this value combinations can be treated as virtual column. The JD statistics can be derived through a virtual column (VC) or just grouping the attribute values directly.

Our contribution can be summarized as:

1) Brings multi-dimensional range selectivity estimation to one dimensional space. It simplifies the multi-dimensional optimal partitioning and optimal sort parameter problems.
2) We don't need to much more worry about the curse dimensionality problem.
3) If it is developed for a set of attributes, then it will work for any subset of it with the same efficiency.
4) When dimensionality increases, unlike MDH, the number of buckets does not increase exponentially. Only a set of one-dimensional PHIST is created. So, the its space complexity is linear on dimentionality.
5) Since the one-dimensional histograms and virtual column are already implemented in commercial databases, so the purposed approach can easily be implemented and used, which increases its practicality usefulness.

2. Problem description and formulation

Let *T* be a relation with *S* attributes and *N* tuples. So, N=|T| rows, X_f ={X_i}, i=$\overline{1,S}$ is the set of attributes of T relation. We are interested in a subset of the X $\subset$ X_f ={X_i}, i=$\overline{1,n}$. For formality we assume that the attribute values are real or integer number. Assume that V_i= {v_i(k): k=1,Di) i=$\overline{1,n}$ is the value set of the X_i attribute. f(v_i(k)) is the frequency or number of tuples of $v_i(k)$ that X_i=v_i(k).

The collection of {v_i(k), f(v_i(k))} pairs is called data distribution of the X_i attribute. The joint frequency f(k_1,k_2,…,k_n) of the value combination < v_1(k_1), .., v_n(k_n,) > is the number of tuples in T that contain v_i(k) in attribute Xi, for all i. The joint data distribution of {X_i}, i=$\overline{1,n}$. is the entire set of *(value combination, joint frequency)* pairs[4]. To estimate multi-dimensional range selectivity, we need to approximate joint data distribution and use it in estimation process.

Suppose that:

$$p_1 = (x_1 = c_1) \wedge (x_2 = c_2) \ldots (x_n = c_n) = \wedge_{i=1}^{n} (x_i = c_i) \qquad (1)$$

$$p_2 = (b_1<x_1<e_1) \wedge (b_2<x_2<e_2) \dots (b_n<x_n<e_n) = \wedge_{i=1}^{n}(C_i) \quad C_i = b_i<x_i<e_i \qquad (2)$$

$$p_3 = (b_1\leq x_1\leq e_1) \wedge (b_2\leq x_2\leq e_2) \dots (b_n\leq x_n\leq e_n) = \wedge_{i=1}^{n}(C_i) \quad C_i = b_i\leq x_i\leq e_i \qquad (3)$$

Here $c_i, b_i \; and \; e_i$ are the given integer or real numbers.

And it is also known that:

$$\mathbb{P}\boldsymbol{r}[(b_i \leq x_i \leq e_i)] = \mathbb{P}\boldsymbol{r}(b_i<x_i < e_i) + \mathbb{P}\boldsymbol{r}(x_i = b_i) + \mathbb{P}\boldsymbol{r}(x_i = e_i).$$

The probability of p3 can be computed based on p1 and p2. Therefore, we mainly focus on p1 and p2.

Problem statement 1

For the given T table and its attributes, it is required to estimate the following queries:

$$Q_1 = \sigma(T)_{c=p1}, \; Q_2 = \sigma(T)_{c=p2} \qquad (4)$$

Estimate $E(Q_1)$ *and* $E(Q_2)$

E()- is the expected cardinality.

Current commercial RDBMS use attribute value independence assumption to estimate selectivity as

$$SEL(p2) = \prod_{i=1}^{n} SEL(C_i). \qquad (5)$$

SEL() is the selectivity of the predicate. The formula (6) is true when the attributes are not correlated.

But, in real world, attribute value assumption is almost wrong and causes large errors in estimation process. So, in fact, if two events are correlated or conditionally depends on each other then:

$P(A \cap B) \neq P(A) * P(B)$, but $P(A \cap B) = P(A/B) * P(B) = P(B/A) * P(A)$ (6)

Problem statement 2

Develop a mechanism that able to estimate $P(A/B)$ *or* $P(B/A)$ *properly or reduce the approximation error.*

That is what the two suggested techniques trying to solve the problem.

2.1 One-dimensional histogram Usage

One-dimensional histogram has been extensively studied and implemented in almost all commercial RDBMS. Its construction quite simple when comparing it to the MDH. In principle, any type of histogram (Equi-Width, Equi-Depth, Maxdiff, V-optimal) can be used, but in this paper, we construct and use compressed type

Equi-Depth histogram. Since compressed histogram provides low errors and its construction cost is smaller than V-optimal or maxdiff, therefore it is a strong competitor to them [1].

The histogram bucket structure we used and implemented in this paper is {(CM_FREQ$_i$,VAL$_i$,FREQ$_i$)} i=$\overline{1, NB}$. NB is number of requested histogram buckets. So, each bucket has three entries. CM_FREQ$_i$ is cumulative frequency, VAL$_i$ is the attribute value, FREQ$_i$ is the frequency of this attribute value. Since VAL$_i$ is the bucket boundary and CM_FREQ$_i$ allow us to have bucket size, so that it is calculated as.

BUCKET_SIZE$_i$ = CM_FREQ$_i$ – CM_FREQ$_{i-1}$.

And the FREQ entry of the histogram allows us to have knowledge at the bucket level frequency distribution. So, we assume that within BUCKET$_i$ we have attribute values distributed uniformly between (VAL$_{i-1}$, VAL$_i$) interval, which will be:

UNIFORM_ROWS$_i$ = CM_FREQ$_i$ – CM_FREQ$_{i-1}$ - FREQ$_i$

And their distinct cardinality – number of distinct values is assumed to be:

NUM_DIST_VALUES_BUCKET$_i$ =VAL$_i$ - VAL$_{i-1}$ + 1

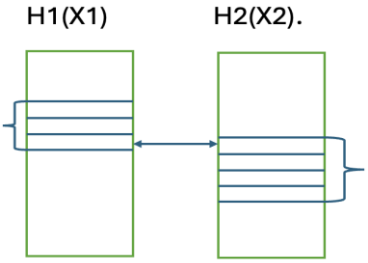
It is very similar structure that is implemented in Oracle DB as hybrid histogram [18]. When histogram is constructed generally there are popular and top-n (or most frequent values) are distinguished. Popular values are in top-n most frequent values, but their individual frequencies are greater than average frequency which is total number of rows in the relation is divided by number of distinct rows. Our histogram algorithm tries to store as much as many popular values explicitly as bucket boundary and this basically depends on the requested number of histogram buckets and number of distinct values. But the bucket might also contain other rows as well. However, ordinary compressed histogram stores the popular and top-n high frequency values in a singleton buckets. As the result, we might get different sized buckets due to popular values. We consider our histogram as compressed histogram and after that, in this paper when we say histogram then it means this is compressed histogram. Now, we can introduce how the selectivity of a predicate for example $b_i<x_i<e_i$ be computed within in a BUCKET$_i$. There are several possible cases depending on b_i,e_i and bucket boundaries. Let's assume that $b_i > VAL_{i-1}$ and $e_i <= VAL_i$, so in this case:

$$\mathbb{P}\mathbf{r}(b_i > VAL_{i-1} \wedge e_i <= VAL_i) = \left(\frac{(BUCKET\_SIZE_i - FREQ_i)(e_i - b_i - \mathbf{1})}{(VAL_i - VAL_{i-1} - \mathbf{1})} + FREQ_i\right)/N$$

It gives us estimation within the bucket. In addition, depending on the query region and histogram bucket boundaries the formula can be adopted accordingly.

2.2 Problem of One-dimensional histograms when estimation selectivity as AVI.

Before going to construct PHIST histogram, let's have a look at why AVI fails with marginal histograms. Let's see that based on two attributes. X_1 and X_2 are given attributes of relation T and H_1 and H_2 are respectively their compressed histograms.



If we compute selectivity based on histograms via simple AVI assumption using (6) like

$$F=P(x1 \in [a,b] \wedge x2 \in [c,d]) \approx P(x1 \in [a,b]) * P(x2 \in [c,d]) \quad (7)$$

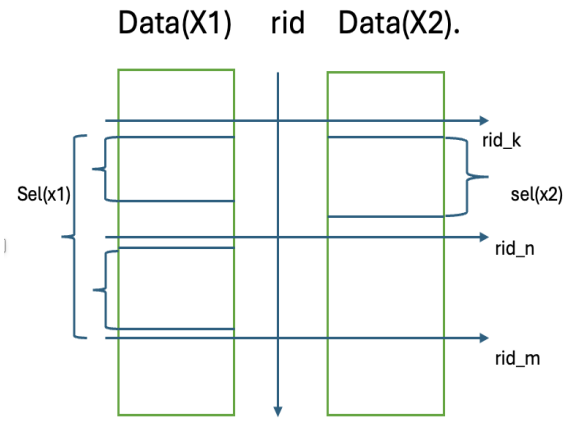
Problem with the formula is that when H1 histogram is created as a compact summary of the X_1 there is not any information or knowledge about X_2 data as the same when H2 histogram is constructed there is no info about X_1. Since the histogram is a compressed or compact summary of the columns, so during this compression and compacting process it is not considered what happens or how it impacts other attribute values as well. They are created independently and as the result there are large errors in estimation process if the attribute values are truly correlated.

The error of the formula depends on how the intersect the attribute values based on their natural order in the relation. So, it is not about how the attribute values distributed independently, it is how the values are distributed together. The histograms of the attributes are computed independently, and the quality of the formula (8) directly depends on the following factors:

1) Properly identifying intersection regions of the X1 and X2 attributes based on natural order during selectivity estimation. Roughly saying, according to the formula (8) we confront two complete histogram each other. It covers the entire relation and corresponding compact data representation.

   {min(X1),max(X1)}<=>{min(X2),max(X2)}.

If a column data represented by a compressed histogram(or any other histogram except *p(F,F)* histograms), then each attribute value is stored in single bucket (or contiguous buckets if the value is popular). Because histogram represents this (one) column data, and the values is ordered and grouped into the buckets. However, the same value itself is distributed over the relation unknowingly. In some parts of the table, they might be clustered and other part scattered, and we compress all of them into one bucket and use it in selectivity estimation. During processing and compressing the data we do not care of other attribute vales as well, which we want to take care of it. Let's consider a popular value and in the below figure its distribution is demonstrated over the table.



If Sel(x1) as computed selectivity for a popular value of X1 and the value in this specific example is distributed between rid_k and rid_m (rid is row identifier in the table or simply rowid) then its estimation whole range (rid_k,rid_m) will be considered. But for sel(x2) it is between (rid_k,rid_n). As a result, we include unproper part of the distribution of the same popular value into the estimation process and it cause the error. The (rid_n, rid_m) part need to exclude and there should be proper counter parts of the attribute values. Naturally, each of such parts, should have its own min, max values and even attribute value and frequency distribution representation as well.

So, two compressed histograms are corresponding each other and only this information is used during selectivity estimation.

2) When P(x1 ∈[a,b]) and P(x2 ∈[c,d]) is computed then relationship of the poplar or top frequent values of the attributes is not considered. So, during selectivity estimation, regardless of predicates and range values of the query, all popular values are used independently. But, depending on the predicates and range values of the attributes, not all popular/top frequent values should be involved or some part of them might need to be involved which indicates attribute correlation that what we need to improve.
3) For above reasons we say the formula is not wrong but instead we say it estimates the selectivity by averagely and could be improved. It can be done by correlating popular/top frequent values of the attributes and having proper parts of corresponding histograms. That is what we introduce PHIST.

Table 1: Notation Table

Notation	Description
H_i	Histogram of X_i domain
n	Number of dimensions
d_i	number of distinct values of the X_i domain
F	function, returns frequency of a value
$V_i(k)$, $k=\overline{1, di}$	attribute values of X_i
$HV_i(k)$, $k=\overline{1, di}$	High frequency values of X_i
NHV_i	Number of high frequency values of X_i
$Vc_i(k)=h(v_1(k), v_2(k)\ldots, v_n(k))$	Virtual column values, h(.) here is a hash function.
HVC(k)	High frequency values of VC
NHVC	Number of high frequency values of VC
dvc	Number of distinct values of VC
B	Number of partitions
L	Number of histogram buckets
E()	Expected cardinality

3. Creating and using PHIST

To explore JD, we need to create histogram for VC or group the value combination of the attributes. After grouping data(or creating histogram for VC)

there can be different kind of distribution styles that we might treat them differently. Let's have a look at few different possible distribution styles and then we will summarize all of them. Note that we ordered the JD data by frequency of the entries by desc.

3.1 If dvc<=L. It means we are allowed to capture and store all entries of JD. Let's call this JD values together as joint Frequency Matrix(JFM).

$v_1(1), v_2(1), \ldots, v_n(1), h(v_1(1)\ldots, v_n(1),), f_1$
$v_1(2), v_2(2), \ldots, v_n(2), h(v_1(2)\ldots, v_n(2),), f_2$
$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$
$v_1(1), v_2(1), \ldots, v_n(1), h(v_1(1)\ldots, v_n(1),), f_n$

In real world the number of distinct values of VC can be quite larger than the given number of buckets. But high frequency values might consume significant part of the JD. let's say *ps* percent which is controlled by users and might have a default value like 99%. Then we still have a chance to estimate both (1) and (2) selectivity. We divide the JD data into two parts as *highly* and *weakly* correlated parts. And the estimations will be:

$$\mathbb{P}\mathrm{r}(p_1) = \mathbb{P}\left[\wedge_{i=1}^{nhvc}(X_i = c_i),\ \wedge c_i \in \{HVC\}\right] + \mathbb{P}\left[\wedge_{i=1}^{nvc-nhvc}(X_i = c_i),\ \wedge c_i \in \{VC\}\backslash\{HVC\}\right] = \frac{f(h(c_1,c_2,\ldots,c_n))}{N} + \frac{N-\sum_{k=1}^{nhvc} f(hvc(k))}{(dvc-nhvc)*N} \quad (8)$$

$$\mathbb{P}(p_2) = \mathbb{P}\left[\wedge_{i=1}^{nhvc}(c_i),\ \wedge c_i \in \{HVC\}\right] + \mathbb{P}\left[\wedge_{i=1}^{nvc-nhvc}(c_i),\ \wedge c_i \in \{VC\}\backslash\{HVC\}\right] \quad (9)$$

The first part of the (12) can be computed as (10), but for the second part since it is considered as not strongly correlated, therefore it could be computed as attribute value independence assumption as:

$$\mathbb{P}\left[\wedge_{i=1}^{nvc-nhvc}(c_i),\ \wedge c_i \in \{VC\}\backslash\{HVC\}\right] = \frac{\prod_{i=1}^{n}[f_{hi}(x_i\in[b_i,e_i]) - f_{hvc}(x_i\in[b_i,e_i])]}{[\prod_{i=1}^{n-1}(N-\sum_{k=1}^{nhvc} f(hvc(k)))]*N} \quad (10)$$

3.2 If dvc>L there is not much enough number of HFV of VC instead just few of them. In other word, all $v_i(\mathbf{k}), \boldsymbol{v_i}(\mathbf{k}) \ldots, \boldsymbol{v_i}(\mathbf{k})$ combinations might have similar height or frequencies. For such kind of distribution, we cannot create JFM directly. Therefore, our suggestion is divide the JD into B buckets and for each bucket we can have individual attribute statistics like minimum, maximum and number of distinct values if needed and so on. Each bucket will have approximately same number of tuples. For example, in bucket B_k for the attribute x_i if minimum and

maximum values are $minx_i$ and $maxx_i$ and if $[b_i,e_i] \subset B_k$ then selectivity for p_2 can be computed as

$$\mathbb{P}(p_2) = \prod \frac{(e_i - b_i)}{maxx_i - minx_i} \quad (11)$$

If query region overlaps or intersects into multiple buckets, then appropriate formulas can be taken. The total estimation will be sum of individual bucket estimations. The problem with this approach is that, here uniform distribution and linearity is assumed. But x_i 's values are distributed in many buckets. To solve this our suggestion is to create individual compressed histograms for each bucket. In total, there will be ***BxnxL*** buckets. Here **n** is the number of attributes and **L** is number of histogram buckets in each bucket. As a result, selectivity estimation within each bucket will be improved. We had relation level marginal histograms and now we have VC bucket level histograms. Suppose in each VC vcb_i bucket there are $|vcb_i|$ rows and there are $(\boldsymbol{hb_j x_i})$ histograms for each x_i , then:

$$\mathbb{P}[b_i < x_i < e_i, b_i, e_i \in \{vcb_j\}] = \frac{f(hb_i x_i)}{|vcb_j|} \Rightarrow \mathrm{E}[\wedge\, b_i < x_i < e_i\,;\; b_i, e_i \in \{vcb_j\}] = \sum_j \frac{1}{|vcb_j|^{n-1}} \prod_{i=1}^{n} f(hb_j x_i) \quad (12)$$

3.4.The Construction Cost, complexity, and Error Measurement

We have mentioned that the SORT parameter of the VC histogram is F. Since it does not have much meaning once we sort it by values. Because we don't investigate predicates like A<VC<B. Therefore, the most needed histogram for VC should be *p(F,F)*, so it could be *compressed* or v-*optimal(F,F)* histogram[1].

We can describe PHIST algorithm simply as below.

1. Scan the table and group attribute value combinations or create virtual column(VC) for combination of attribute values.
2. Sort it by frequency in descending order.
3. Partition the sorted data into **B** number of partitions with equal number of rows.
4. For each partition construct appropriate a set of one – dimensional histograms with **L** buckets
5. Maintain JFM for each partition when it is allowed and store them in the database catalog. Or maintain the JFM at global level and correlate or distribute the JFM with the PHIST, so that, which entries of JFM belong to which histogram.

6. Estimate selectivites using above formulas.

Currently to simplify the construction we materialize individual partitions, however, the case of without materialization, will be look at later[do that using some streaming or quantile-based approaches, link please]. Despite to confront two independent histograms in formula (8), now we confront a pair of set of coreresponding histograms. So, when selectivity estimation instead of $\{H_1\}<=>\{H_2\}$, we use $\{H_{1i}\}<=>\{ H_{2i} \}$, $i=\overline{1,n}$..

To create histogram, the challenging part is sorting and as it was described in MDH construction[3], its sorting cost is $\boldsymbol{Z} * \mathbf{log}[\frac{\boldsymbol{Z^n}}{\boldsymbol{B}^{\frac{n(n-1)}{2}}}]$ here **Z** is number of pages in relation T and **n** is number of dimensions. Due to the delimator, it indicates that it does not do the complete **n** times of sorting. In our approach, we do group of the data, then partition it and then create individual atribute level one-dimensional histograms, which requires n times of sorting. But, there are two advantages. First of all, usually after groupng data its size is reduced significantly. Lets $\boldsymbol{GBZ}$ be the number of pages after grouong the data, then usualy **GBZ << Z** and the sort cost will be $\boldsymbol{GBZ} * \mathbf{log}(\boldsymbol{GBZ^n})$. And the total grouping and sorting together . $\boldsymbol{Z} * \mathbf{log}[\mathbf{Z}] + \boldsymbol{GBZ} * \mathbf{log}(\boldsymbol{GBZ^n})$. Secondly, after creating PHIST we don't to have create individual marginal histograms for each dimension separately using n times sorting again. Because we already have PHIST as a set of histograms for each dimension. That is why they can be easily derived from PHIST by aggregating and merging them[link]. So, PHIST also provides marginal histograms as well. Moreover, the algorithm can be parallelized easily , partition level as well.

***Lemma 1 (Construction Time)**. The PHIST is constructed in **O(**$\boldsymbol{Z}(\mathbf{1} + \boldsymbol{log}[\boldsymbol{Z}]) + \boldsymbol{GBZ} * \boldsymbol{log}(\boldsymbol{GBZ^n})$**)**. It includes one scan of the T table then grouping data based on n attributes and then creating individual histograms.*

***Lemma 2 (Query Time)**. The algorithm computes an estimate in **O(n*B*L)**, where n is data dimensionality, B is number of VC buckets and L is number of marginal histogram buckets.*

4. **Creating and using RHIST.**

Firstly, in PHIST we focused on grouping of attribute values and tried to approximate the JD. The next question is how we can improve existing histogram-based AVI approach. Because histograms on single columns have been studied very well and effectively implement in almost all commercial DBMS. For example, if there are X_1, X_2, X_3 attributes then how the formula (7) can be refined? In other

words, if A,B and C is selectivity of X_1, X_2, X_3 accordingly then how can we provide better estimation for P(A/B), P(B/A), P(A/C), P(C/B)?

The idea is that when a histogram is created for one of the attributes for example X_1 then for each histogram bucket of the X_1 the appropriate data of the rest of attributes also can be processed and as a result, for each bucket of X_1 we will have appropriate corresponding rest of attributes data. X_1 is called primary attribute, and its histogram buckets are called primary buckets. It will allow us to better estimate conditional probabilities. To process the appropriate data of X_2, X_3 attributes, streaming or sketching algorithms can be used.

So, for each HB_i bucket of X_1 attribute there will be corresponding CB_iX_2 and CB_iX_3 data of X_2, X_3 attributes. Here, HB_i bucket of X_1 provides maximum and minimum values, number of tuples in the bucket and individual frequency for its boundary as provided the bucket structure in the section **2.1**. But, how the CB_iX_2 and CB_iX_3 data should be look like, how it can be organized, summarized and which information it should contain to support conditional probability. In our current approach we consider that each of CB_iX_2 and CB_iX_3 data contains minimum, maximum values of X_i attributes, their frequencies if possible, and allowed top-n the most frequent values with their frequencies. So, for each CB_iX_i should contain the following information.

{MINV, MAXV, TOP_N_VALUES, FREQUENCIES} or

Its structure will be A list of {VALUE,FREQUENCY} pairs which already contains minimum, maximum, and allowed top_n values along with their frequencies if it is detected.

For example, by default if the allowed top-n values for each such CB_iX_i data is 1 then each of it will contain 6 number: minimum, maximum and one top frequent value and their frequencies. We call them as corresponding buckets(CB). Number of tuples is not necessary to be stored in them, since it is already represented in primary attribute's histogram buckets.
The estimation formula in each CB will be:

$$\mathbb{P}[b_i < x_i < e_i, b_i, e_i \in CX_iB_i] = \frac{(e_i - b_i - nr_tfc)\left(|cx_ib_i| - \sum_{k=1}^{nr_tfc} f\left(tfv_{cx_ib_i}(k)\right)\right)}{max_{cx}b_i - max_{cx}b_i - nr_tfc} + \sum_{k=1}^{nr_tfc} f\left(tfv_{cx_ib_i}(k)\right) \quad (13)$$

Here $\boldsymbol{nr_tfc}$ is number of top freq. values meet the given condition, $\boldsymbol{|cx_ib_i|}$- size of the bucket, and $\boldsymbol{tfv_{cx_ib_i}(k)}$ **i**s the top frequent values. So based on (1**3**), we calculate the selectivity by using top frequent values and uniform distribution by

excluding them. To create CB, we used space-saving algorithm[22,23], however a streaming or any other algorithm could be used, like count-min sketch with priority queues.

A question can be raised that which attribute should be selected as primary attribute here. For that we need to define error and to see how it can be reduced by selected attributes.

Lemma3 (the maximum error). The maximum error in selectivity estimation for a query region that intersects one primary bucket is

$$Max\_Err_i[b_i < x_i < e_i, b_i, e_i \in CX_iB_i] = Max[\frac{1}{2B}, \frac{T/S - \sum_{k=1}^{nr_tfc} f\left(tfv_{cx_ib_i}(k)\right)}{2(T/S - nr_tfc)}] * n \quad (19)$$

And the total err of the whole query region will be:

$$Max\_Err = \sum_{i=1}^{p} Max\_Err_i \quad (20)$$

P – is the query region that it covers all buckets.

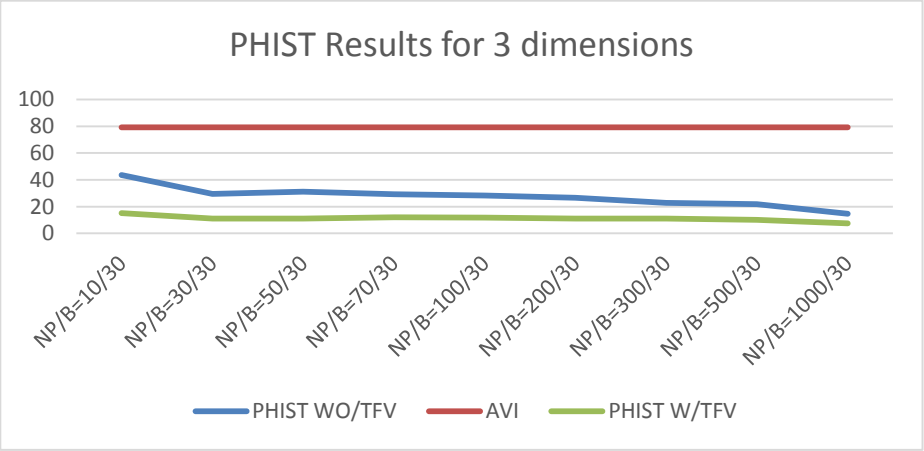
Lemma 4 (Construction Time). RHIST is constructed in $O_{io}(Z(1 + log[Z]))$*+Ocpu(n*N)*.

Lemma 5 (the query time). The algorithm computes an estimate in O(p*n), where n is data dimensionality, p is the query region.

According to the lemma 4 the maximum estimation error depends on the number of buckets and the number of allowed top-n frequent values for each CB. The error can be reduced by increasing number of primary histogram buckets and to record more top-frequent values. However, for the given top-n values we can reduce the error by increasing the number of buckets of the primary attribute. It gives and general idea that primary attribute could be selected the column that has maximum number of distinct values. When number of buckets increased then relationship between attribute values become clearer. But, to improve accuracy of this approach we also can keep the top-n the most frequent values of the attribute combination what we did for PHIST. In each bucket, the selectivity for top-n most frequent values of the attribute combination and that for rest of the tuples can be computed separately and then their sum will give us the final selectivity.

5. Experimental Evaluation

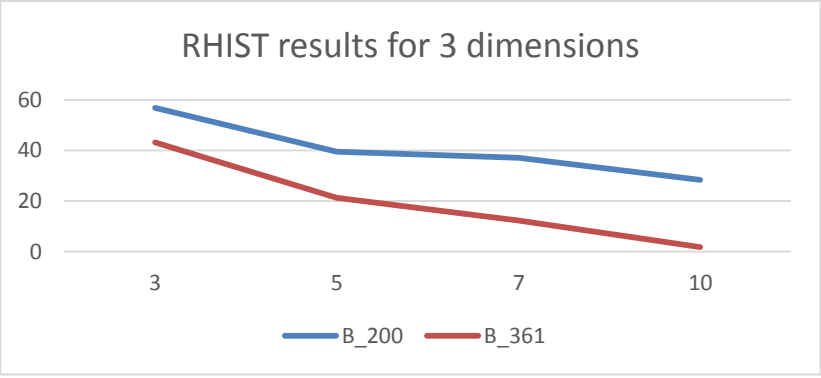
We used forest cover dataset [10] from UCI ML repository with three attributes: *aspect, slope and slope*.



As it is seen AVI based estimation gives 80% relative error and increasing number of buckets might not improve accuracy immediately. With 10 Equi-depth partitions and 10 individual partition buckets the relative error is 12.93 percent, which is good, however partitions from 10 to 500 there is no significant improvement, we see the improvement only after having more than 2000 partitions. The problem is that so far, we don't maintain the JFM for each partition yet. The accuracy depends on how many top frequent values of JFM entries we keep and use.

Experimental evaluation for RHIST.

We tested the same range query as used in previous example.



References

[1] V.Poosala,Y.E.Ioannidis,P.J.Haas,andE.J.Shekita. “Improved Histograms for Selectivity Estimation of Range Predicates”.

[2] Y. Ioannidis . The History of Histograms(abridged)

[3] Muralikrishna M., DeWitt D.: Equi-Depth Histograms for Estimating Selectivity Factors for MultiDimensional Queries. SIGMOD Conf. (1988) 28-36

[4] Poosala V., Ioannidis Y.: Selectivity Estimation Without the Attribute Value Independence Assumption. VLDB Conf. (1997)

[5] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In SIGMOD, pages 13–24,

[6] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes, VLDB J., 14(2):137–154, 2005.

[7] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional selectivity estimation using compressed histogram information. In SIGMOD, pages 205–214, 1999.

[8] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In SIGMOD, pages

[9] M. Shekelyan, A. Dignos, and J. Gamper, “Digithist: a histogram- ¨ based data summary with tight error bounds,” PVLDB, vol. 10, no. 11, pp. 1514–1525, 2017.

[10]UCI machine learning repository. https://archive.ics.uci.edu/ml/index.ph

[11] V. Leis, B. Radke, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. Query optimization through the looking glass, and what we found running the join order benchmark. The VLDB Journal, 27(5), 2018.

[12] Khanna S, Muthukrishnan S, Patterson M (1998) On approximating rectangle tiling and packing.In: Proceedings of the 9th annual symposium on discrete algorithms (SODA), San Francisco, January 1998

[13] Y. E. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. ACM Trans. Database Syst., 18(4):709–748, 1993.

[14] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In VLDB, volume 98, pages 24–27, 1998.

[15] Muthukrishnan S, Poosala V, Suel T (1999) On rectangular partitionings in two dimensions: algorithms, complexity, and applications.In: Proceedings of the ICDT 1999, Jerusalem, January 1999, pp 236–256 27.Olken F, Rotem D

(1990) R

[16] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. J. Comput. Syst. Sci., 52(3):550–569, 1996

[17] 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

[18] https://docs.oracle.com/en/database/oracle/oracle-database/18/tgsql/histograms.html

[19] JAGADISH, H. V., KOUDAS, N., MUTHUKRISHNAN, S., POOSALA, V., SEVCIK, K. C., AND SUEL, T. 1998. Optimal histograms with quality guarantees. In Proceedings of the VLDB Conference, 275–286

[20] SUDIPTO GUHA University of Pennsylvania NICK KOUDAS University of Toronto and KYUSEOK SHIM Seoul National University. Approximation and Streaming Algorithms for Histogram Construction Problems. ACM Transactions on Database Systems, Vol. 31, No. 1, March 2006, Pages 396–438.

[21] Ioannidis, Y.E.; Poosala, V.(1995). “Balancing histogram optimality and practicality for query result size estimation”. Proceedings of the 1995 ACMS SIDMOD international conference on Management of data – SIGMOD’95. P.233

[22] Graham Cormode Marios, Hadjieleftheriou. "Finding Frequent Items in Data Stream" Proceedings of the VLDB Endowment, Volume 1, Issue 2 Pages 1530 – 1541, 2008

[23] SpaceSaving$^{\pm}$: an optimal algorithm for frequency estimation and frequent items in the bounded-deletion model Fuheng Zhao, Divyakant Agrawal, Amr El Abbadi, Ahmed Metwally, PVLDB, 15(6): 1215 - 1227,2022