Improving Range Query Result Size Estimation Based on a New Optimal Histogram

Wissem Labbadi and Jalel Akaichi

Computer Science Department, ISG-University of Tunis, Le Bardo, Tunisia
wissem.labbadi@yahoo.fr, jalel.akaichi@isg.rnu.tn

Abstract. Many commercial relational Data Base Management Systems (DBMSs) maintain histograms to approximate the distribution of values in the relation attributes and based on them estimate query result sizes. A histogram approximates the distribution by grouping data into buckets. The estimation-errors resulting from the loss of information during the grouping process affect the accuracy of the decision, made by query optimizers, about choosing the most economical evaluation plan for a query. In front of this challenging problem, many histogram-based estimation techniques including the equi-depth, the v-optimal, the max-diff and the compressed histograms have well contributed to approximate the cost of a query evaluation plan. But, most of the times the obtained estimates have much error. Motivated by the fact that inaccurate estimations can lead to wrong decisions, we propose in this paper an efficient algorithm, called Compressed-V2, for accurate histogram constructions. Both theoretical and effective experiments are done using benchmark data set showing the promising results obtained using the proposed algorithm. We think that this algorithm will significantly contribute for helping to solve the problem of Multi-Query Optimization (MQO) resulting from queries interactions especially in Relational Data Warehouses (RDW) which represent the ideal environment in which complex OLAP queries interact with each other.

Keywords: Optimal histograms, Query result size estimation, Intermediate query result distribution, DBMS, Estimation error, Multi-query optimization, Query interaction.

1 Introduction

Many commercial DBMSs maintain a variety of types of histograms to summarize the contents of the database relation by approximating the distribution of values in the relation attributes and based on them estimate sizes and value distributions in query results [1, 2, 3, 4]. Different techniques for constructing histograms are described in [5]. The simplest approach for constructing a histogram on attribute X is by partitioning the domain D of X into β (β >1) mutually disjoint subsets called buckets. A histogram approximates the distributions by grouping the data values into buckets. This grouping into buckets loses information. This loss of information engenders errors in estimates based on these histograms. The resulting estimation-sizes errors

H.L. Larsen et al. (Eds.): FQAS 2013, LNAI 8132, pp. 40–56, 2013.

directly or transitively affect the accuracy of the decision, made by query optimizers, about choosing the most efficient access plan for a query [6] and undermine the validity of the optimizers' decisions.

The problem of query optimization consists in choosing, among many different query evaluation plans, the most economical one for a given query. Since the number of query evaluation plans increases exponentially with the number of relations involving the query [7], query optimization was becoming a worthwhile problem. A query can be performed by means of different intermediate operations such as join. A simple sequence of join operations that leads to the same final result is called a query evaluation plan [7]. Each query evaluation plan has an associated cost which depends on the number of operations performed in the intermediate joins. In [8], it has shown that errors in query result size estimates may increase exponentially with the number of operations performed in the intermediate joins. In worse of the cases, the chance of choosing the optimal query evaluation plan decreases since the query optimizer uses erroneous data to accomplish its task [9]. In that case, the query optimizer must estimate various parameters for the intermediate results of the operations and then use the obtained values to estimate the corresponding parameters of the results of subsequent operations [9]. Even if the original errors are small, their transitive effect on estimates derived for the final result may be devastating and so leading query optimizers to wrong decisions. For multi-join queries that are processed as a sequence of many join operations, the transitive effect of error propagation among the intermediate results on the estimates derived for the complete query may be destructive. This problem has been solved by approximating the cost of a query evaluation plan using histogram-based estimation techniques including the equi-width [10], the equi-depth [11], the v-optimal [1, 9, 12, 13], the max-diff [3] and the compressed histograms [3]. The idea was to estimate the query result sizes of the intermediate results and based on them selecting the most efficient and economical query evaluation plan.

Another important problem in which query result estimation techniques may be very useful is the phenomenon of query interaction which raises the problem of multiple queries optimization (MQO) especially in the relational data warehouse context (RDW). Relational data warehouses represent the ideal environment in which complex OLAP queries interact with each other. The problem of MQO combines the problem of efficient buffer management and the problem of query scheduling [14, 15]. It consists in finding an optimal scenario of queries processing that permits a total benefits from the buffered intermediate results which represents a major cause of performance problems in database systems. In fact, before executing a given query, it may get benefit from the actual content of the buffer if it has some intermediate results with previous queries [16]. Based on this scenario, if the query scheduler has a snapshot of the buffer content (intermediate results), it may reorder the queries to allow them getting benefit from the buffer [16].

Motivated by the fact that inaccurate estimations can lead to wrong decisions, our contribution can be summarized on preparing an experimental comparative study of the effectiveness of the different optimal histograms reported in the literature in order to identify the best one for reducing error in the estimations of sizes and value

distributions especially in the results of queries with high complexity, e.g., multi-join queries. We envisage by this study to determine the main features of a good histogram in order to take them into account when developing our algorithm called Compressed-v2 algorithm for accurate histogram construction. Both theoretical and effective experiments are done using real data sets.

This paper is organized as follows. Section 2 provides an overview of several earlier and some more recent classes of histograms that are close to optimal and effective in many estimation problems. In section 3, we propose a new technique based on an effective algorithm called HistConst to construct a very promising histogram called compressed-v2 in terms of query result size estimation accuracy. In section 4, we propose a running example to show the efficiency of our algorithm. Section 5 presents a set of experiments to compare the effectiveness of the different histogram. Finally, Section 6 concludes and outlines some of the open problems in this area.

2 State of the Art

The buckets in a histogram are determined according to a partitioning rule and are limited by the disk space. We classify, in this section, the histograms listed in the literature into two classes based on two partitioning constraints. The first constraint consists in partitioning the attribute domain based on trivial rules and it concerns earlier histograms like the equi-width [10] and the equi-depth [11] histograms. The second constraint aims to avoid grouping vastly different values into the same bucket and it covers relative recent histograms like v-optimal [1, 9, 12, 13], max-diff [3] and compressed histograms [3].

2.1 Earlier Histograms

Trivial Histograms. This kind of histograms has a single bucket where all the attribute values fall into the same bucket. Frequencies approximated based on this histogram are identical for all attribute values [17]. This histogram assumes the uniform distribution over the entire attribute domain [6] and this assumption, however, didn't hold in real data. That's why trivial histograms usually have large error rate in query result estimation [8, 18].

Example 1. Let us consider the histogram maintaining, over one single bucket, the approximated frequencies of the attribute SALARY in a relation R with information on 100 employees (see Fig. 1). The domain of SALARY is the interval from 1000 $ to 5000 $.

According to the histogram in Fig. 1, the number of employees having for example a salary equal to 1500 $, denoted SEL (SALARY=1500 $) [11], is approximated by the average frequency of all salaries which is $S(R)/v(R, SALARY)$ where *S (R)* is the size of the relation R and *V(R, SALARY)* is the number of distinct values present in the attribute SALARY. Three different approaches were proposed in the literature to approximate the number of distinct values within a bucket [1, 6, 11].

The accurate number of tuples satisfying the above query is anywhere from 0 to 100. So, this approximation can be wrong at least by 50% (for *V(R, SALARY)* = 2) and in general usually by more than 50% (for *V(R, SALARY)* > 2) which represents a very large error rate.

Equi-Width Histograms. This kind of histograms consists in dividing the domain of the attribute values into K equal-width buckets and counting the number of tuples falling into each bucket [11]. Typically, equi-width histograms have 10 to 20 buckets [10].

Example 2. Let us consider the histogram maintaining the distribution of the attribute SALARY from example 1 (see Fig. 2). For reasons of simplicity, let this histogram divide the domain of the attribute into 3 equal-width buckets.

Continuing with the same query from example1, the accurate number of the employees having **SALARY = 1500$** is anywhere from 0 to 48. So the true percentage of the employees with **SALARY = 1500 $ is** anywhere from 0 to 0.48 ($0 \leq$ SEL (=1500 $) $\leq$0.48). An estimation, on average, of SEL (=1500 $) from the histogram in Fig. 2 corresponds to the mid-point in this range which is 0.24. So, this estimate can be wrong by 0.24. In general, the maximum error in estimating SEL (= *Const*) on average, denoted SEL~ (=Const) [11], is half the height of the bucket in which *Const* falls.

In [11], it has been shown that estimations of histograms belonging to the class of *equi-width* histograms are often better than *trivial* ones. They have frequently large errors since they force buckets to have equal width without controlling the height of each bucket. In such a histogram, we may find too high buckets and too low other ones. This huge disparity is due to the unexpected distribution of values over the entire attribute. In general, the distributions of values in the attributes of relations rarely follow any functional description, such as Zipf distribution [19] which leads to an inequitable distribution of values over the different buckets. In that case, if the bucket in which *Const* falls is too high, the range in which SEL (=*Const*) belongs will be very large (the superior limit of the range is close to 100%) and a selectivity estimate will be wrong by 50% (mid-point in this range).

We can conclude that in order to control the maximum estimation error, the height of each bucket in the histogram should be controlled. Hence, the idea of creating histograms having buckets with equal height instead of equal width.

Equi-Depth Histograms. The maximum error in estimating from a histogram the selectivity of comparison, based on relational operators, is half the height of the bucket in which the comparison constant falls into. This error can be very close to 0.5, with an unlucky distribution of attributes values, where the tallest bucket contains almost 100% of the tuples in the relation. Creating a histogram where the attribute values are equally distributed over the different buckets will avoid having, in all cases, large errors in selectivity estimates. Such a histogram is called equi-depth [11]. In an equi-depth histogram, called also equi-height, the sum of the frequencies in each bucket is the same. This kind of histograms guarantees estimation with small error (usually < 0.5) and the maximum error can be reduced to an arbitrarily small value by increasing sufficiently the number of buckets in a way that half the height of a bucket

will be negligible. For the construction of this histogram, we must first sort the attribute values in an ascending order to obtain a height balanced histogram.

Example 3. The distribution of the salaries of 100 employees in an equi-sum fashion over 3 equal height buckets is represented in Fig. 3.

Again choosing the maximum error in selectivity estimates as the half of the bucket, the estimation of SEL (=1500$) can be wrong at most by 0.16 (half the height of the first bucket in which 1500 falls). This error is 1 time and a half less than the error that can be present in the selectivity estimate obtained using an equi-width histogram. But the difficulty in this type of histograms consists in how to determine the required boundaries of the buckets in order to guarantee the equality of height between the different buckets.

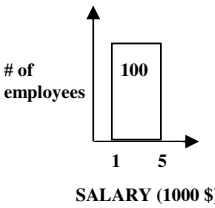

Fig. 1. Distribution of salaries over one singleton bucket

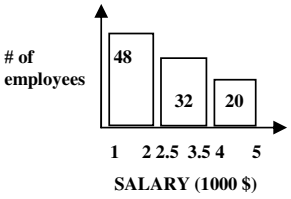

Fig. 2. Distribution of salaries over equal-width buckets

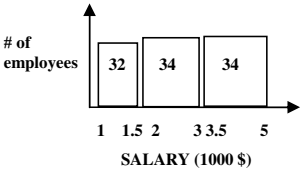

Fig. 3. Distribution of salaries over equal-height buckets

2.2 Relative Recent Histograms

V-Optimal Histograms. The v-optimal histograms [9, 12, 13], called also variance-optimal try to avoid grouping vastly different values into a bucket by reducing the weighted variance between the actual and the approximate distribution over all the approximated values within each bucket [13]. This variance is defined as $\sum_{j=1}^{\beta} p_j V_j$, where p is the number of frequencies, V is the variance of frequencies in the jth bucket and β is the maximum number of buckets.

The v-optimal histogram is optimal for estimating on average the result sizes of equality join and selection queries [1]. In order to approximating the number of distinct values with in a bucket, contrary to the previous histograms which instead of storing the actual number of distinct values in each bucket, they make assumptions about it such as continuous values assumption and point value assumption and both can lead to significant estimation errors, V-optimal histograms record every distinct attribute value that appeared in each bucket. Since bucket groups close frequencies and under the above assumption, all frequencies will be close to the average of frequencies so that estimations will be close to the actual results.

Definition 1. *Let H1 and H2 be two different histograms partitioning the values of an attribute X into the same number β (≥1) of buckets. The v-optimal histogram on X, among H1 and H2, is the histogram with the least variance.*

Example 4. Fig. 4 illustrates the above definition by the meaning of two different histograms H1 and H2 on the attribute SALARY with 3 buckets in each one. The frequencies of the different attribute values are listed in Fig. 4. a Fig. 4. b and Fig. 4. c show the partitioning technique employed for grouping the frequencies into buckets respectively for H1 and H2.

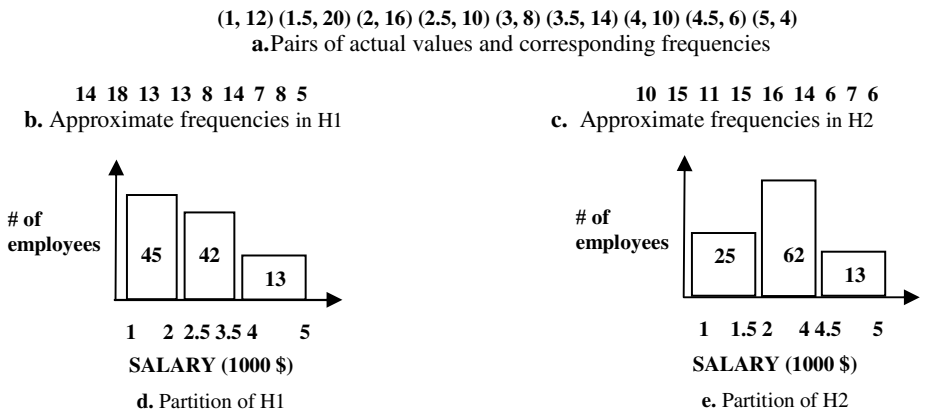

Fig. 4. Example of optimal histograms

The cumulated variances V_{H1} and V_{H2} respectively of H1 and H2 are calculated as follows. V_{H1} = 14+74+16.6667 = **104.6667** and V_{H2} = 24.5+52.8+0.5 = **77.8**. Based on Definition 1, the v-optimal histogram on the attribute SALARY, defined previously, between H1 and H2 is H2 since it has the least cumulated variance.

Maxdiff Histograms. The maxdiff histograms try to avoid grouping vastly different values within a bucket by inserting a boundary between two adjacent values v_i and v_{i+1} if the difference between the area of v_{i+1} and v_i is one of the β-1 largest such differences [3]. The area a_i of v_i is defined as $a_i = f(v_i).s_i$ where s_i is the spread of v_i and is defined as $s_i = v_{i+1} - v_i$ [3, 20]. Continuing with the set of values shown in Fig. 4a, the differences between the areas of the different successive values, noted Δ area, are calculated in Table 1. So, according to this table the bucket boundaries of a maxdiff histogram, approximating the distribution of values in Table 1 over 3 buckets, are inserted respectively between the two pairs of adjacent values (2, 2.5) and (3, 3.5) since they differ the most than the other pairs of adjacent values. The corresponding Maxdiff histogram is illustrated in Fig. 5.

This histogram estimates the number of tuples having the value 1500 in the attribute SALARY to be 22 engendering then an error on average that can reach 22%.

The comparison between the different histograms based on the error obtained in the estimates provided by each one for the same query (SEL (SALARY = 1500)) shows that v-optimal and max-diff are significantly more accurate and practical than earlier histograms.

Table 1. Computing the spread, area and Δ area

Value	1	1.5	2	2.5	3	3.5	4	4.5	5
Frequency	12	20	16	10	8	14	10	6	4
Spread	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	-
Area	6	10	8	5	4	7	5	3	-
Δ Area	4	2	3	1	3	2	2	-	-

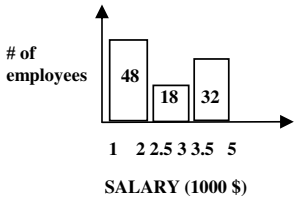

Fig. 5. Distribution of salaries in a Maxdiff histogram

Compressed Histograms. The compressed histograms try to achieve the new partition constraint consisting at avoiding to group, into a bucket, values with highly different frequencies by selecting the n values having the highest frequencies and placing them separately in n singleton buckets. The remaining values are partitioned over equi sum buckets [3]. Different techniques have been proposed to determine either a value is one of the n highest values or not. For example, in [3] they choose n to be the number of values that exceed the sum of the total frequencies divided by the number of buckets.

The DBMS maintaining a compressed histogram estimates accurately the selectivity each time the query looks for the periodicity of a high frequent value.

Example 5. Let's consider a compressed histogram approximating the distribution of the salaries over 5 buckets (see Fig. 6). According to [3] to choose the highest values, 1500$ is considered a high frequent value and is stored separately in a singleton bucket.

The compressed histograms by keeping values with high frequencies in singleton buckets and grouping contiguous values into buckets, they achieve great accuracy in estimating selectivity in databases [3]. That's, this histogram provides an accurate estimation on average (with a null error) of the same previous query SEL (SALARY = 1500$).

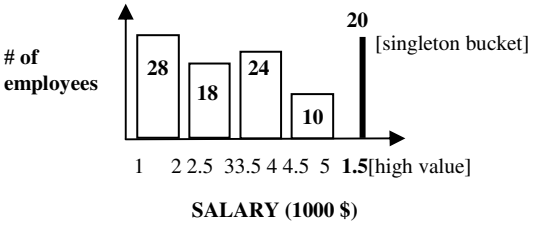

Fig. 6. Distribution of salaries in a compressed histogram

3 Compressed-V2 Histogram

The problem of constructing a good histogram and maintaining it well is primordial for the validity of the query optimizers' decisions [8, 18]. Due to their typically low-error estimates and simplicity in representing data distributions in low costs, there has

been considerable work on identifying good histograms for estimating the result sizes of various query operators with reasonable accuracy [1, 3, 11, 21, 22]. The proposed histograms differ in how the attribute values are assigned to buckets to achieve good estimates and especially by the error rate in their estimates. In this work, we propose an efficient algorithm for constructing an improved version, called *compressed-v2*, of existing *compressed* histogram [3].We developed both theoretical and effective experiments to underline the effectiveness and the accuracy of our algorithm and to prove that the new version of *compressed* histogram generates the lowest estimation error among the existing techniques.

In a *compressed-v2* histogram, the *n* highest attribute values are stored separately in *n* singleton buckets. In our algorithm, we choose *n* to be the number of values that exceed the sum of all values divided by the number of buckets. The rest of values are partitioned over *maxdiff* buckets [11] instead of being partitioned over *equi-depth* ones [3]. An optimization phase is applied to the exceptional buckets in order to guarantee they generate good estimations. An exceptional bucket is a *maxdiff* bucket taller than the *equi-depth* bucket(s) approximating both the same distribution of values.

The problem of multi-query processing consists in finding an optimal scenario of query processing that permits a total benefits from the buffered intermediate results which represents a major cause of performance problems in database systems. The effectiveness of our histogram in estimating the size and the distribution of the intermediate results helps to well ordering the queries in order to allow them to get benefit form the buffer.

3.1 Definitions and Problem Formulation

In this section we define the accuracy of a histogram and formulate the problem studied in this paper.

Definition 2. *Let H1, respectively H2 be a compressed, respectively compressed-v2 histograms approximating the frequency distribution of an attribute X. We say that H2 is more optimal than H1 if and only if the error of H2 in approximating the frequency of each infrequent value of X is strictly less than the error of H1 in approximating the frequency of the same value.*

Theorem 1. Given a frequency distribution of a data set, a max-difference bucket with a height h1 provides estimation on average more accurate than an equi-depth bucket with a height h2 for all h1 ≤ h2.

Proof. Consider a relation R containing an attribute X. The value set V of X is the set of values of X that are present in R and F the set of their corresponding frequencies. Let M and E be respectively a maxdiff and an equi-depth histograms constructed by partitioning the values of V into β (≥1) buckets.

Let $(h_i^M)_{i=1..N}$ and $(h_i^E)_{i=1..k}$ be the respective heights of the buckets $(B_i^M)_{i=1..N}$ and $(B_i^E)_{i=1..k}$ that compose respectively the histograms *M* and *E*.

Let's take a maxdiff bucket B_i^M and an equi-depth bucket B_j^E, having common values that lie in their ranges, such that $h_i^M \leq h_j^E$ for a given $1 \leq i \leq N$ and $1 \leq j \leq K$. To prove that the frequency approximations on average of the common values based on the bucket B_i^M are more accurate than those based on the bucket B_j^E, it suffices to prove that: Error $(B_i^M) \leq$ Error (B_j^E), where Error(B_i^M), respectively Error (B_j^E) represents the total error of the approximation of B_i^M, respectively of B_j^E.

This inequality is verified since M, the max-diff histogram, is already constructed by minimizing the difference between the grouped values, whereas equi-depth permits vastly different values to be stored in the same bucket. Thus, the values grouped in B_i^M are close to the average of frequencies in B_i^M while those in B_j^E are dispersed from the average of frequencies in B_j^E. Hence, Error $(B_i^M) \leq$ Error (B_j^E).

The case where $h_i^M > h_j^E$ and there are common approximated values between B_i^M and B_j^E for $1 \leq i \leq N$ and $1 \leq j \leq K$ represents the main problem we focus in this paper. We try to improve the accuracy of these$(B_i^M)_{i=1..N}$, called exceptional buckets, using the proposed ***HistConst algorithm***.

3.2 HistConst Algorithm

In general, the construction of a histogram on an attribute is performed on two steps. The first consists on partitioning the frequencies of the attribute into buckets, and the second step is to approximate the frequencies and values in each bucket in some technique [2]. We suggest in this section a naïve algorithm called **HistConst** which gives an accurate histogram with respect to the estimation error specified for the given sequence of values and number of buckets in O (n) time. The **HistConst** algorithm is illustrated in Fig. 7.

HistConst Algorithm. This algorithm takes in input the approximate frequencies of the attribute values and the number of permitted buckets. The **HistConst** algorithm proceeds as follow. First, there will be a call to the procedure **Find( )** to determine the highest values to store them separately in singleton buckets. Then, the procedure **maxdiff( )** takes care to partition the rest of values, over the remaining buckets in a maxdiff fashion, by inserting a bucket boundary between two adjacent values that differ the max. In the optimization phase, we try to reduce the height of the exceptional buckets to guarantee accurate estimations. This phase proceeds as follows:

We consider the height of an equi-depth bucket as a threshold.

Migrate, from each *exceptional bucket*, the *minimum values* in their order in the bucket range to the *previous bucket* while the height of this latter is lower than the threshold and the height of the *exceptional bucket* remains greater than the threshold. Once the previous bucket reach the threshold and the exceptional bucket is still higher than the threshold, then migrate all possible *maximum values* in the bucket range to the *next bucket* without that this latter exceeds the threshold.

Values from the previous bucket (respectively next bucket) can be migrated, if necessary, in their turn to its previous (respectively its next) bucket in order to respect the maximum tolerated height for a bucket.

Algorithm HistConst
Objective: Construct an optimal histogram with respect to the estimation error specified for the given sequence of values and number of buckets.
Input:B:Number of permitted buckets (b_1, b_2, …,b_B)
threshold: maximum tolerated height for a bucket
check: booleanvariable that receive True if the actual bucket is an exceptional one.
Output: compressed-v2 histogram
begin
1. Find (F, V,B, V')
2. Maxdiff(L, F, B', maxdiff)

Optimization phase

3. **Repeat**
   check :=false
4. **For**i := 1 **to**B'**do** {
5. If (exceptional_bucket(b_i)) then {
6. check:=true
7. **While** (h(prev_bucket(b_i)) < threshold) and (h(b_i) > threshold) **do** { // *h(b_i):determines the height of b_i*
8. migrate (min_val(b_i), b_i, prev_bucket(b_i)) // *min_val(b_i):determinesminimum value in the range of b_i*
9. }
10. If h(prev_bucket(b_i) ≥ threshold) then // *prev_bucket(b_i): determines the previous bucket of b_i*
11. **While** (h(next_bucket(b_i)) < threshold) and (h(b_i) > threshold) **do** { // *next_bucket(b_i): determines the successive bucket of b_i*
12. migrate (max_val($_{bi}$), $_{bi}$, next_bucket($_{bi}$)) //*max_val(b_i):determines maximum value in the range of b_i*
13. }
14. }
15. }
16. **Until** (check = false)
17. Result:return compressed-v2

end

Fig. 7. The HistConst algorithm

Finding Highest Values. We present in the Fig. 8 a pseudo code to find the high frequent values among those actually present in the relation.

Procedure Find (F, V, B, **V'**)
Inputs:V: set of values of the attribute that are present in the relation, V= {v_i | 1 ≤ i ≤ N}
F: frequency vector of the attribute, F= {f(v_i) | 1 ≤ i ≤ N}
Output: V': set of the high frequent values, V' = {v_i | f(v_i) > $\frac{\sum f(vi)}{B}$, 1 ≤ i ≤ N}
begin
1. **for**i: = 1**to** N**do**{
2. if (f(v_i) >$\frac{\sum f(vi)}{B}$) then
3. Add(v_i, V')
4. }
5. Result: return V'

end

Fig. 8. Code of the procedure Find

Having the approximated values and the corresponding approximated frequencies, the procedure **Find( )** takes care to determine the highest values to store them separately in singleton buckets. Each value is compared to the sum of all source values divided by the number of total buckets. If the value exceeds this quotient, then is considered a high frequent value.

Constructing Maxdiff Histogram. After finding the highest values and storing each one separately in a singleton bucket, we propose an algorithm of the procedure maxdiff presented in Fig. 9 to partition the remaining values following the technique that consists in separating vastly different values into different buckets.

Procedure Maxdiff(L,F, B',**maxdiff**)
Inputs:L:set of the remaining values (low frequent values) that are present in the relation, L= $\{v_j \mid f(v_j) \leq \frac{\sum f(vj)}{B}, 1 \leq j \leq M < N\}$
B':number of the remaining buckets (non singleton buckets) for grouping the low frequent values
Output:maxdiff:max-diff histogram partitioning the remaining values
begin
1. **for**i:= 1**to**(B'-1)**do** {
2. [max_area := 0]**for** j :=1 **to** (M-1) **do** {
3. Δ Area := $[f(v_{j+1})*S_{j+1}] - [f(v_j)*S_j]$
4. If (Δ Area >max_area) then {
5. max_area := Δarea
6. bound :=j
7. }
8. }
9. *Insert bucket_boundary* (v_{bound}, $v_{bound+1}$)
10. }
11. Result: return maxdiff
end

Fig. 9. Code of the procedure maxdiff

The procedure **Maxdiff** begins first by calculating the differences between all the adjacent values. Then, it inserts bucket boundaries between the pairs of adjacent values that differ the most in their frequencies with respect to the number of buckets permitted to partition the remaining values.

Procedure Exceptional_bucket(b^M)
Input:b^M:maxdiff bucket
begin
1. If (h (b^M) > threshold) then
2. Exceptional_bucket:= True
3. ElseExceptional_bucket := False
4. Result: return Exceptional_bucket
end

Fig. 10. Code of the function Exceptional_bucket

Finding Exceptional Buckets. We propose in this section a boolean function to determine the exceptional buckets (Fig. 10). We remind that the threshold is the height of an equi-depth bucket partitioning the same values which is approximately equal to $\frac{\sum f(vi)}{B'}$ with $1 \leq i \leq M$. We remind also that B' is the number of buckets used to partition the remaining values.

After approximating the low frequent values and their frequencies in each bucket according to the maxdiff partitioning technique, this function returns true for each exceptional bucket, false else. In the affirmative, the corresponding bucket undergoes the change of the optimization phase described above in order to adjust its height with respect to the height of an equi-depth bucket.

In Table 2., we describe the complexity of the HistConst main components.

Table 2. HistConst time complexity

Algorithm	Time Complexity
Procedure Find	O(N)
Procedure Maxdiff	O(M)
Algorithm HistConst	O(N)
Procedure Exceptional_bucket	O(B')

Where N is the number of attribute values, M is the number of low frequent values and B' is the number of non-singleton buckets.

4 Running Example

Suppose we have the following values from an integer-valued attribute with their corresponding low frequencies (see Table 3).

Table 3. Set of integer values with their frequencies

Value	1	2	3	4	5	6	7	8	9
Frequency	2	1	2	3	4	1	2	3	2
ΔFrequency	1	1	1	1	3	1	1	1	

Following the *HistConst* algorithm steps to partition these values over four buckets as in a maxdiff histogram, the three pairs of adjacent values that differ the most in their frequencies are (5, 6), (2, 3) and (7, 8). Thus, the bucket boundaries are placed between these adjacent values (see Fig. 11).

According to *HistConst* algorithm, the bucket with a range [3, 5] and a height equal to 9 is considered an exceptional bucket. In the context of approving exceptional buckets in a compressed-v2 histogram, the procedure Exeptional_bucket( )migrates the value 3 from the second bucket to the first bucket, as shown in Fig. 12, since the two adjacent values 2 and 3 are contiguous and grouping them into the same bucket doesn't affect the accuracy of frequency approximation inside a bucket.

The result of the optimization phase is a histogram that discards vastly different values and then partitions them like in an equi-depth histogram such that the sum of frequencies in each bucket is approximately the same. This is in contrast to the equi-depth histograms that permit vastly different values to be stored in the same bucket. The resulting histogram is illustrated in Fig. 13.

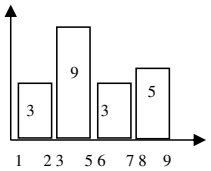

Fig. 11. Partitioning of low frequent values in a maxdiff fashion

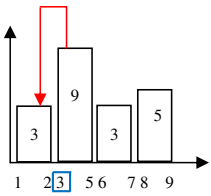

Fig. 12. Improvement phase

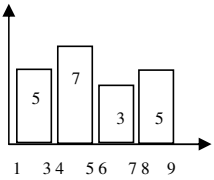

Fig. 13. Partitioning the rest of values after optimization

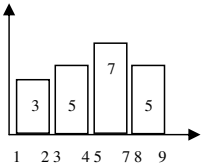

Fig. 14. Partitioning infrequent values in equi-depth fashion

4.1 Selectivity Estimation of Low Frequent Values with Compressed and Compressed-v2 Histograms

The accuracy of estimates of range query result sizes obtained through maintained histograms depends heavily on the partitioning rules used to group attribute values into buckets [9, 11]. Here, we compare the average errors incurred when estimating the selectivity only of low frequent values based on compressed and compressed-v2 histograms. The same highest values are chosen for the two types of histograms and hence their frequencies are similarly approximated in both histograms.

The compressed histogram approximating, over four buckets, the frequencies of these values is illustrated in Fig. 14. To investigate the accuracy of query result size estimates obtained from compressed and compressed-v2 histograms, we choose to compare the accuracy of the selectivity estimation of the values 5 and 6 obtained from the two types of histograms described as follows:

- ***Compressed***: The value 5 falls in the third bucket (Fig. 14). Then, SEL (SALARY = 5) is estimated by the average of frequencies in this bucket which is 2. The true fraction of tuples with salary equal to 5 is 4, then this estimate is wrong by 0.5. Similarly, the value 6 falls into the same bucket and its selectivity is estimated on average by 2. The true fraction of tuples with salary equal to 6 is 1 and hence this estimate is wrong by 0.5.
- **Compressed-v2:** The value 5 falls in the second bucket (Fig. 13) and SEL (SALARY = 5) is estimated by 3. The true fraction of tuples with salary equal to 5 is 4 which mean that the estimate is wrong by 0.25. Contrary to the compressed histogram, the value 6 is separated from the value 5 and is stored in the third bucket (Fig. 13) since they are judged as two large different values. SEL (SALARY = 6) is estimated on average by 1 where the real frequency is 1. The error in the estimate in this case is equal to 0.

Comparing the errors in the estimates based on the two types of histograms, we see clearly that the compressed-v2 approximates much better the frequency of the values 5 and 6 than the compressed histogram. The frequencies of the other values are almost equally approximated in the two histograms since each group contains contiguous values which are stored in approximately equal height buckets in both histograms.

5 Experimentation Results

We investigated the effectiveness of the different histogram types cited above for estimating range query result sizes. The average errors due to the different histograms, as a function of the number of buckets, are computed each time when estimating based on histograms the result size of a selection query where the selectivity conditions are related each time to values with different frequencies like infrequent values, balanced values and very frequent values.

The experiments were conducted on six different histogram algorithms including equi-width, equi-depth, v-optimal, maxdiff, compressed and compressed-v2 in three specified histogram data-frequency category including low, balanced and very high, while using two different distributions of the attribute Salary from the *American League Baseball Salaries (Albb)* and *National League Baseball Salaries (Nlbb)* databases respectively for the years 2003 and 2005. The values in the attribute Salary vary from 300 000$ to 22 000 000$ in the two databases.

The frequencies of the values in the attribute Salary in the database of *Albb* (respectively Nlbb) vary from 1 to 44 (respectively from 1 to 25). We consider [1..9] (respectively [1..8]) to be the range of low frequencies, [10..30] (respectively [9..15]) the range of balanced frequencies and [31..44] (respectively [16..25]) to be the range of very high frequencies.

In the following experiment, we studied the performance of the different histograms by comparing through several graphs the typical behavior of the histograms errors in approximating the frequencies of different values with varying the number of buckets. For an efficient study of the effectiveness of these histograms, we select randomly three values from each database: an infrequent value, a balanced value and a very frequent value. The errors in approximating the frequency of a given value are represented in a graph separately. The x-axis of each graph shows the number of buckets and the y-axis shows the average error of each histogram for different number of buckets.

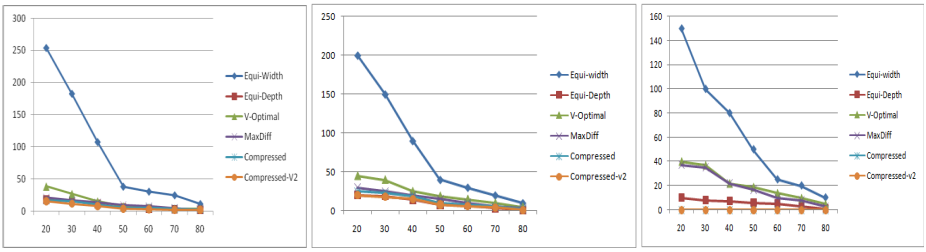

Fig. 15. Average error as a function of the number of buckets of approximating, in Nlbb, the frequency of a) an infrequent value, b) a balanced value, and c) a very frequent value

We select from the database of Nlbb the following values with their corresponding real frequencies (1000000, 1), (500000, 10) and (316000, 25). The errors of the six histograms in approximating the frequencies of these values are illustrated respectively in Fig. 15.a, Fig.15.b and Fig. 15.c.

We select from the database of Albb the following values with their corresponding real frequencies (7500000, 1), (600000, 10) and (300000, 25). The errors of the six histograms in approximating the frequencies of these values are illustrated respectively in Fig. 16.a, Fig.16.b and Fig. 16.c.

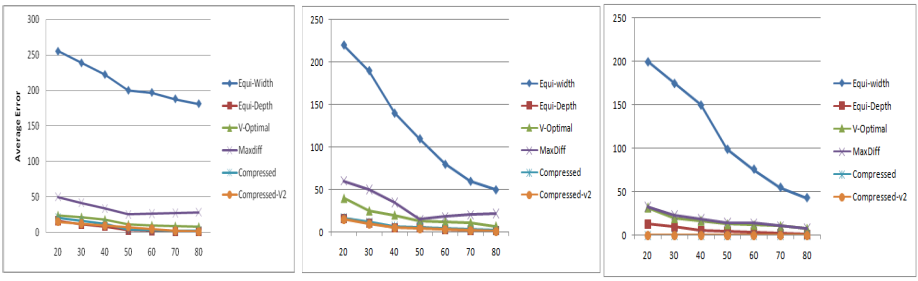

Fig. 16. Average error as a function of the number of buckets of approximating, in Albb, the frequency of a) an infrequent value, b) a balanced value, and c) a very frequent value

Looking at the different figures, we observe that the error generated is monotony proportional to the number of buckets. As shown in the two figures, the accuracy can be reached when increasing the number of buckets for all histogram types and the compressed-v2, compressed, max-diff and v-optimal histograms are significantly better than the others that they show the least error for different number of buckets. Moreover, the equi-width histogram exhibits the worst accuracy.

Based on the different figures, we distinguish clearly, by comparing the average errors generated by the entire set of histograms when estimating the selectivity of infrequent values, balanced or very frequent values, a set of effective histograms, i.e. compressed-v2, compressed, V-optimal and Max-diff, where the compressed-v2 presents each time the least approximation error for different number of buckets. The same behavior of compressed-v2 errors in all the figures improves the victory of this histogram over the other ones that it gives 100% accurate approximation of the frequencies the highest values since their actual frequencies are stored separately in individual buckets.

In conclusion, the comparison between the different histograms presented above based on the average error generated when estimating range query result sizes shows that the histograms based on the new partition constraints and on their heads the compressed-v2 performs always significantly better than those based on trivial constraints.

6 Conclusion and Future Work

The problem of minimizing the error in estimating range query result sizes remains a real challenge despite the serious research done on identifying classes of optimal

histograms that generate least errors in the estimations of sizes and value distributions especially in the results of queries with high complexity, e.g., multi-join queries.

In this paper, we provided an overview of several earlier and some more recent classes of histograms that are close to optimal and effective in many estimation problems. In addition to that, we have introduced a new algorithmic technique, HistConst, for constructing a more accurate histogram called compressed-v2. An experimental comparative study was proposed to study the effectiveness of the different classes of optimal histograms reported in the literature and our proposed histogram in estimating sizes and value distributions especially in the results of complex queries, e.g., multi-join queries. The experiments show that estimations based on our histogram are always better than those based on the other remaining types of histograms.

The identification of the optimal histogram remains an open field. As several new research opportunities appear, we will try to identify optimal histograms for different types of queries to limit not only the average estimation error but also other metrics of error, to determine the appropriate number of buckets to build the optimal histogram and to find the histogram that can handle uncertain data.

An important direction for research is to focus on the problem of data stream which is the transmission of the flow of data that changes over time. Existing database systems do not process data streams efficiently and this makes this area a popular search field [23, 24].

References

1. Ioannidis, Y., Poosala, V.: Balancing histogram optimality and practicality for query result size estimation. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 233–244 (1995)
2. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K., Suel, T.: Optimal histograms with quality guarantees. In: Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York, USA, pp. 275–286 (1998)
3. Poosala, V., Ioannidis, Y.E., Haas, P.J., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 294–305 (1996)
4. Jagadish, H.V., Jin, H., Ooi, B.C., Tan, K.-L.: Global optimization of histograms. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 223–234 (2001)
5. Yu, C., Philip, G., Meng, W.: Distributed top-N query processing with possibly uncooperative local systems. In: Proc. 29th VLDB Conf., Berlin, Germany, pp. 117–128 (2003)
6. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: Proceedings of the ACM SIGMOD International Symposium on Management of Data, Boston, Mass., pp. 23–34 (June 1979)
7. John Oommen, B., Rueda, L.G.: An empirical comparison of histogram-like techniques for query optimization. In: Proceedings of the 2nd International Conference on Entreprise Information Systems, Stafford, UK, July 4-7, pp. 71–78 (2000)

8. Ioannidis, Y., Christodoulakis, S.: On the propagation of errors in the size of join results. In: Proceedings of the 1991 ACM SIGMOD Conference, Denver, CO, pp. 268–277 (May 1991)
9. Ioannidis, Y., Christodoulakis, S.: Optimal histograms for limiting worst-case error propagation in the estimates of query optimizers. To appear in ACM-TODS (1992)
10. Kooi, R.P.: The optimization of queries in relational databases. PhD thesis, Case Western Reserver University (September 1980)
11. Shapiro, G.P., Connell, C.: Accurate Estimation of the Number of Tuples Satisfying a Condition. In: Proceedings of ACM-SIGMOD Conference, pp. 256–276 (1984)
12. Ioannidis, Y.: Universality of serial histograms. In: Proceedings of the 19th Int. Conf. on Very Large Databases, pp. 256–267 (December 1993)
13. Poosala, V., Ioannidis, Y.: Estimation of query-result distribution and its application in parallel-join load balancing. In: Proceedings of the 22nd Int. Conf. on Very Large Databases, pp. 448–459 (1996)
14. Gupta, A., Sudarshan, S., Viswanathan, S.: Query scheduling in multi query optimization. In: IDEAS, pp. 11–19 (2001)
15. Thomas, D., Diwan, A.A., Sudarshan, S.: Scheduling and caching in multi query optimization. In: COMAD, pp. 150–153 (2006)
16. Kerkad, A., Bellatreche, L., Geniet, D.: Queen-Bee: Query interaction- aware for buffer allocation and scheduling problem. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 156–167. Springer, Heidelberg (2012)
17. Ioannidis, Y.: Query optimization. In: ACM Computing Surveys, Symposium Issue on the 50th Anniversary of ACM, vol. 28, pp. 121–123 (1996)
18. Christodoulakis, S.: Implications of certain assumptions in database performance evaluation. ACM TODS 9(2), 163–186 (1984)
19. Zipf, G.K.: Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology. Addison-Wesley, Cambridge (1949)
20. Liu, Y.: Data preprocessing. Department of Biomedical, Industrial and Human Factors Engineering Wright State University (2010)
21. Ioannidis, Y., Poosala, V.: Histogram-based solutions to diverse database estimation problems. IEEE Data Engineering Bulletin 18(3), 10–18 (1995)
22. Muralikrishna, M., Dewitt, D.J.: Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In: Proceedings of ACM SIGMOD Conference, pp. 28–36 (1988)
23. Mousavi, H., Zaniolo, C.: Fast and Accurate Computation of Equi-Depth Histograms over Data Streams. In: Proceedings of EDBT, Uppsala, Sweden, March 22-24 (2011)
24. Gomes, J.S.: Adaptive Histogram Algorithms for Approximating Frequency Queries in Dynamic Data Streams. In: 12th International Conference on Internet Computing, ICOMP 2011, Las Vegas, NV, July 18-21 (2011)