Estimating Range Queries using Aggregate Data with Integrity Constraints: a Probabilistic Approach

Francesco Buccafurri[1], Filippo Furfaro[2], Domenico Saccà[3]

[1] DIMET, University of Reggio Calabria, 89100 Reggio Calabria, Italy, bucca@ing.unirc.it
[2] DEIS, University of Calabria, 87030 Rende, Italy, furfaro@si.deis.unical.it
[3] ISI-CNR & DEIS, 87030 Rende, Italy, sacca@unical.it

Abstract. In fast OLAP applications it is often advantageous to provide approximate answers to range queries in order to achieve very high performances. A possible solution is to inquire summary data rather than the original ones and to perform suitable interpolations. Approximate answers become mandatory in situations where only aggregate data are available. This paper studies the problem of estimating range queries (namely, sum and count) over aggregate data using a probabilistic approach for computing expected value and variance of the answers. The novelty of this approach is the exploitation of possible integrity constraints about the presence of elements in the range that are known to be null or non-null. Closed formulas for all results are provided, and some interesting applications for query estimations on histograms are discussed.

1 Introduction

Traditional query processing deals with computing exact answers by possibly minimizing response time and maximizing throughput. However, a recent querying paradigm, on-line analytical processing (OLAP) [11, 13, 5], often involves complex range queries over very large datacubes (i.e., multidimensional relations with dimension and measure attributes) so that the exact answer may require a huge amount of time and resources. As OLAP queries mainly deals with operations of aggregation (e.g., count and sum) of the measure values on dimension ranges, an interesting approach to improve performances is to store some aggregata data and to inquiry them rather than the original data thus obtaining approximate answers — this approach is very useful when the user wants to have fast answers without being forced to wait a long time to get a precision which often is not necessary.

The possibility of returning approximate answers for range queries has been first explicitly addressed in [12] but, in that case, the approximation is temporary since results are output on the fly while the tuples are being scanned and, at the end, after all original tuples are consulted, the user will eventually get the correct answer.

The issue of computing approximate range query answers by never accessing original tuples but only consulting aggregate data has very recently started receiving a deal of attention. Typical approaches consist in re-using statistical techniques which have been applied for many lusters inside query optimizers for selectivity estimation [17]. We recall that three major classes of statistical techniques are used for selectivity estimation: *sampling*, *histograms* and *parametric modeling* — see [2] for a detailed survey. Interesting applications of sampling and histogram techniques already exist, see for instance [8, 9] and [10], respectively. The usage of sampling techniques are also used for approximate join-queries answering [1]. A recent technique for selectivity estimation, wavelet-based histograms, has been already applied to approximate answering of range queries [18].

In this paper we propose a probabilistic approach to compute approximate answers to range queries (in particular, count and sum queries) by consulting a compressed representation of the datacube, that is a partition of the datacube into blocks of possibly different sizes storing a number of aggregate data (number of non-null tuples and sum of their measure values) for each block. Our approximated results will come with a detailed analysis of the possible error so that, if the user is not satisfied with the obtained precision, s/he may eventually decide to submit the query on the actual datacube. In this case, it is not necessary to run the query over all tuples but only on those portions of the range that do not fit the blocks.

Our approach is not concerned with the problem of finding the most effective compressed representation of a datacube to increase accuracy in query estimation — instead this is the main goal of the sampling and histogram techniques mentioned above. We are involved with the "apparently" simpler problem of performing interpolation of aggregate data once the compressed representation for the datacube has been decided. This means that our approach can be also used to interpolate data from summarized ones for which detail tuples are not available. This case has been first studied in [6] and interesting results have been obtained by enforcing the optimization of some criterion like the smoothness of the distribution of values. Our approach does not make any assumption on data distribution and perform estimations extending the probabilistic framework introduced in [3].

The novelty of our approach is that we exploit additional information on a datacube that is often available under the form of integrity constraints. In particular, we assume the existence of constraints stating that a minimum number of null or non-null tuples are present in given ranges. Such a situation often arises in practice. For instance, given a datacube whose dimensions are the time (in terms of days) and the products while the measure is the amount of daily product sales, realistic integrity constraints are that the sales are null during the week-end while at least 4 times a week the sales are not null.

In the paper we analyze two types of integrity constraints:

- *number of elements that are known to be null:* we are given a function $LB_{=0}$ returning, for any range R, a lower bound to the number of null tuples

occurring in D — so $|R| - LB_{=0}(R)$ is an upper bound on the number of non-nulls occurring in the range;
- *number of elements that are known to be non-null:* we are given a function $LB_{>0}$ returning, for any range R, a lower bound for the number of non-null tuples occurring in R.

The two functions $LB_{=0}$ and $LB_{>0}$ are assumed to be monotone, to require a little amount of additional storage space and to be computable very efficiently — actually in time constant w.r.t. the size of the compressed datacube. Possible future research directions could explore other types of integrity constraints: then the problem of interpolating data from compressed representation could eventually enter the field of knowledge discovery and data mining. This explain why we have above stressed that the problem is only apparently simple.

Our problem is therefore the following: given a range R inside a datacube block B for which we know the count t (the number of non-null tuples) and the sum s of their measure values, we want to compute the estimation (mean and variance) of the count t_R and the sum s_R for the range R, knowing that $(|R| - LB_{=0}(R)) \leq t_R \leq LB_{>0}(R)$ and $(|\bar{R}| - LB_{=0}(\bar{R})) \leq t_{\bar{R}} \leq LB_{>0}(\bar{R})$, where $\bar{R}$ is the range in B complementary to R.

The results we provide are formulas for mean and variance of both count and sum queries; besides the formulas are closed so that they can be computed very efficiently. For instance, suppose that the block B has size 120, the number of non-nulls in it is 80 and their sum is 12000 and that the range R consists of the first 30 tuples in the block. Without integrity constraints we have that the expected value for s_R is obviously $(30/120) \times 12000 = 3000$ — note that the knowledge about the number of non-nulls does not contribute to the estimation. Suppose now that we know that at least 4 of the first 20 tuples in the block are null but at least 2 of them are not null; moreover, at least half of the last 20 tuples are not null. Thus, $LB_{=0}(R) = 4$, $LB_{>0}(R) = 2$, $LB_{=0}(\bar{R}) = 0$ and $LB_{>0}(\bar{R}) = 10$. By applying our formulas we now obtain that the expected value for s_R is 4100.

Estimating mean values is not enough in most situations: we also need to compute the possible error in the estimation. For instance, given a block of size 100 and sum 10000 and given a range R coinciding with half of the block, the expected value for the sum in R is independent from the number of non-nulls in B. But it is obvious that the error in the case this number is 2 is much higher than for the case with, say, 90 non-nulls; so we need to consult the variance of the estimation before concluding that it is meaningful. Our results include closed formulas also for the variance of both count and sum queries. Indeed the proofs of such formulas are rather long and complex so we have included only one proof in the appendix. Besides, for reason of space, all the other proofs are either left out or only sketched.

The paper is organized as follows. In Section 2 we introduce the compressed representation of a datacube and the integrity constraints about the number of null or non-null tuples in the datacubes ranges. In Section 3 we fix the probabilistic framework for estimating *count* and *sum* range queries on a datacube M by means of random queries variables over the population of all datacubes

which both have the same aggregate data as M and satisfies the integrity constraints. For the sake of the presentation, in Section 4 we perform range query estimation for the simple case that only integrity constraints about the null tuples are available (i.e., only the function $LB_{=0}$ is given); the general case is treated in the subsequent section. Finally, in Section 6 we give some interesting applications of our formulas for the estimation of frequency distribution inside a bucket of a histogram [14–16]. The most common approach is the *continuous value assumption* [17]: the sum of frequencies in a range of a bucket is estimated by linear interpolation. We shall show that this computation does not yield a correct estimation for the case of bucket whose extremes (or at least one of them) is known to be not null. This situation, that arises for many of the most popular histogram representations, can be formalized in terms of our integrity constraints, thus obtaining more accurate estimations as well as the evaluation of their errors.

2 Compressed Datacubes and Integrity Constraints

Let $\mathbf{i} = <i_1, \ldots, i_r>$ and $\mathbf{j} = <j_1, \ldots, j_r>$ be two r-tuples of cardinals, with $r > 0$. We extend common operators for cardinals to tuples in the obvious way: $\mathbf{i} \leq \mathbf{j}$ means that $i_1 \leq j_1, \ldots i_r \leq j_r$; $\mathbf{i}+\mathbf{j}$ denotes the tuple $<i_1+j_1, \ldots, i_r+j_r>$ and so on. Given $p \geq 0$, $\mathbf{p}^r$ (or simply $\mathbf{p}$, if r is understood) denotes the r-tuple of all p. Finally, $[\mathbf{i}..\mathbf{j}] = [i_1..j_1, \ldots, i_r..j_r]$ denotes the range of all tuples from $\mathbf{i}$ to $\mathbf{j}$, that is $\{\mathbf{q} |\ \mathbf{i} \leq \mathbf{q} \leq \mathbf{j}\}$.

A *multidimensional relation* R is a relation whose scheme consists of $r > 0$ *dimensions* (also called *functional attributes*) and $s > 0$ *measure attributes*. The dimensions are a key for the relation so that no two tuples have the same dimension value. For the sake of presentation but without loss of generality, we assume that

- $s = 1$ and the domain of the unique measure attribute is the set of cardinals, and
- $r \geq 1$ and the domain of each dimension q, $1 \leq q \leq r$, is the range $[1..n_q]$, where $n_q > 2$, i.e., the projection of R on the dimensions is a subset of $[\mathbf{1}..\mathbf{n}]$, where $\mathbf{n} = <n_1, \ldots, n_r>$.

Given any range $[\mathbf{i}..\mathbf{j}]$, $\mathbf{1} \leq \mathbf{i} \leq \mathbf{j} \leq \mathbf{n}$, we consider the following *range queries* on R:

- *count query*: $count^{[\mathbf{i}..\mathbf{j}]}(R)$ denotes the number of tuples of R whose dimension values are in $[\mathbf{i}..\mathbf{j}]$, and
- *sum query*: $sum^{[\mathbf{i}..\mathbf{j}]}(R)$ denotes the sum of all measure values for those tuples of R whose dimension values are in $[\mathbf{i}..\mathbf{j}]$.

Since the dimension attributes are a key, the relation R can be naturally viewed as a $[\mathbf{1}..\mathbf{n}]$ matrix (i.e., a *datacube*) M of elements with values in $\mathcal{N}$ such that for each $\mathbf{i} \in [\mathbf{1}..\mathbf{n}]$, $M[\mathbf{i}] = v$ if the tuple $<\mathbf{i}, v>$ is in R or otherwise $M[\mathbf{i}] = 0$ — then $\mathbf{i}$ is a *null element* if either $<\mathbf{i}, 0>$ is in R or no tuple with dimension value $\mathbf{i}$ is present in R. The above range queries can be now re-formulated in terms of array operations as follows:

- $count^{[\mathbf{i}..\mathbf{j}]}(R) = count(M[\mathbf{i}..\mathbf{j}]) = |\{\mathbf{q}|\ \mathbf{q} \in [\mathbf{i}..\mathbf{j}] \text{ and } M[\mathbf{q}] > 0\}|$;
- $sum^{[\mathbf{i}..\mathbf{j}]}(R) = sum(M[\mathbf{i}..\mathbf{j}]) = \sum_{\mathbf{q}\in[\mathbf{i}..\mathbf{j}]} M[\mathbf{q}]$.

We next introduce a *compressed representation* of the relation R by dividing the datacube M into a number of blocks and by storing a number of aggregate data for each of them. To this end, given $\mathbf{m} = <m_1, \ldots, m_r>$ in $[\mathbf{1}..\mathbf{n}]$, a $\mathbf{m}$-*compression factor* for M is a tuple $F = <f_1, \ldots, f_r>$, such that for each q, $1 \leq q \leq r$, f_q is a $[0..m_q]$ array for which $0 = f_q[0] < f_q[1] < \cdots < f_q[m_q] = n_q$. For each $\mathbf{k} = <k_1, \ldots, k_r>$ in $[\mathbf{1}..\mathbf{m}]$, let $F^+(\mathbf{k})$ and $F^-(\mathbf{k})$ denote the tuples $<f_1[k_1], \ldots, f_r[k_r]>$ and $<f_1[k_1 - 1] + 1, \ldots, f_r[k_r - 1] + 1>$, respectively . Therefore, F divides the range $[\mathbf{1}..\mathbf{n}]$ into $m_1 \times \cdots \times m_r$ blocks $B_{\mathbf{k}}$, one for each tuple $\mathbf{k} = <k_1, \ldots, k_r>$ in $[\mathbf{1}..\mathbf{m}]$; the *block* $B_{\mathbf{k}}$ has range $[F^-(\mathbf{k})..F^+(\mathbf{k})]$ and size $(f_1[k_1] - f_1[k_1 - 1]) \times \cdots \times (f_r[k_r] - f_r[k_r - 1])$ if $\mathbf{k} > \mathbf{1}$ or $f_1[k_1] \times \cdots \times f_r[k_r]$ otherwise.

For instance, consider the $[1..10, 1..6]$ matrix M in Figure 1(a), which is divided into 6 blocks as indicated by the double lines. We have that $\mathbf{m} = <3, 2>$, $f_1[0] = 0$, $f_1[1] = 3$, $f_1[2] = 7$, $f_1[3] = 10$, and $f_2[0] = 0$, $f_2[1] = 4$, $f_2[2] = 6$. The block $B_{<1,1>}$ has size 3×2 and range $[1..3, 1..4]$; the block $B_{<1,2>}$ has size 3×2 and range $[1..3, 5..6]$, and so on.

1	3	0	0	5	5
5	1	0	4	5	9
5	5	0	2	5	0
2	0	0	0	1	5
5	3	0	0	5	5
5	0	0	1	5	5
0	1	0	1	5	0
1	0	0	0	5	5
1	1	0	0	6	2
1	0	0	0	2	3

(a)

([1..3,1..4],8,26)	([1..3,5..6],5,29)
([4..7,1..4],7,18)	([4..7,5..6],7,31)
([8..10,1..4],4,4)	([8..10,5..6],6,23)

(b)

Fig. 1. A two-dimensional datacube and its compressed representation

A *compressed representation* of the datacube M consists of selecting a $\mathbf{m}$-compression factor F and storing the following aggregate data on the F-compressed blocks of M:

- the $[\mathbf{1}..\mathbf{m}]$ matrices $M_{count,F}$ and $M_{sum,F}$ such that for each $\mathbf{k} \in [\mathbf{1}..\mathbf{m}]$,

$$M_{cs,F}[\mathbf{k}] = cs(M[F^-(\mathbf{k})..F^+(\mathbf{k})])$$

  where cs stands for *count* or *sum*;

The compressed representation of the datacube M in Figure 1(a) is represented in Figure 1(b) by a matrix of triples, one for each block; the values of each triple

indicates respectively the range, the number of non-null elements and the sum of the elements in the corresponding block. For instance, the block $B_{<1,1>}$ has range $[1..3, 1..4]$ and 8 non-null elements with sum 26; the block $B_{<1,2>}$ has range $[1..3, 5..6]$ and 5 non-null elements with sum 29, and so on.

We assume that the we are given a compressed representation of a datacube M as well as additional information on M under the form of integrity constraints on the content of M. The representation of such constraints needs a little amount of additional storage space; besides, the requirements defined by the constraints are expressed in terms of aggregate data which are computed by suitable function in constant time — thus the functions are not dependent on the actual contents of M. As discussed in the Introduction, data distributions often match this property in real contexts. For instance, consider the case of a temporal dimension with granularity *day* and a measure attribute storing the amount of sales for every day. In this case, given any temporal range, it is easily recognizable a number of *certain* null values, corresponding to the holidays occurring in that range. In such cases, the constraints provide additional information that can be efficiently computed with no overhead in terms of storage space on the compressed representation of M.

Let $2^{[\mathbf{1}..\mathbf{n}]}$ be the family of all subsets of indices in $[\mathbf{1}..\mathbf{n}]$. We analyze two types of integrity constraints:

- *number of elements that are known to be null:* we are given a function $LB_{=0} : 2^{[\mathbf{1}..\mathbf{n}]} \rightarrow \mathcal{N}$ returning, for any D in $2^{[\mathbf{1}..\mathbf{n}]}$, a lower bound to the number of null elements occurring in D; the datacube M satisfies $LB_{=0}$ if for each D in $2^{[\mathbf{1}..\mathbf{n}]}$, $\sum_{\mathbf{i}\in D} count(M[\mathbf{i}]) \leq |D| - LB_{=0}(D)$, where $|D|$ is the number of elements of M in D;
- *number of elements that are known to be non-null:* we are given a function $LB_{>0} : 2^{[\mathbf{1}..\mathbf{n}]} \rightarrow \mathcal{N}$ returning, for any D in $2^{[\mathbf{1}..\mathbf{n}]}$, a lower bound for the number of non-null elements occurring in D; the datacube M satisfies $LB_{>0}$ if for each D in $2^{[\mathbf{1}..\mathbf{n}]}$, $\sum_{\mathbf{i}\in D} count(M[\mathbf{i}]) \geq LB_{>0}(D)$.

The two functions $LB_{=0}$ and $LB_{>0}$ are monotone: for each D', D'' in $2^{[\mathbf{1}..\mathbf{n}]}$, if $D' \subset D''$ then both $LB_{=0}(D') \leq LB_{=0}(D'')$ and $LB_{>0}(D') \leq LB_{>0}(D'')$.

Suppose that $LB_{=0}([4..6, 1..3]) = 3$ and $LB_{>0}([4..6, 1..3]) = 1$ in our runnning example. Then we infer that the number of non-null elements in the range $[4..6, 1..3]$ is between 1 and $(6-4+1) \times (3-1+1) - 3 = 6$. Note that the compressed representation of M in Figure 1(b) only says that the block $[4..7, 1..4]$ has 7 non-nulls; so, from this information, we only derive that the bounds on the number of non-null elements in $[4..6, 1..3]$ are 0 and 7.

3 The Probabilistic Framework for Range Query Estimation

We next introduce a probabilistic framework for estimating the answers of range queries (*sum* and *count*) by consulting aggregate data rather than the actual datacube. To this aim, we consider the queries as random variables and we

give their estimation in terms of mean and variance. More precisely, a range query Q on a given datacube M is estimated by a random query variable $\overline{Q}$, defined by applying Q on a datacube $\tilde{M}$ extracted from the population of all datacubes, whose compressed representations is 'compatible' with the one of M. Thus, the estimation of the range query Q is only based on the knowledge of the compressed representation of M. A crucial point in such estimation is the definition of population 'compatible' with the compressed representation of the given datacube M.

We start from the population of the datacubes having the same aggregate data that we assume available for M: $M^{-1}_{cs,F}$ is the set of all the $[\mathbf{1}..\mathbf{n}]$ matrix M' of elements in $\mathcal{N}$ for which both $M'_{count,F} = M_{count,F}$ and $M'_{sum,F} = M_{sum,F}$. We next restrict the population $M^{-1}_{cs,F}$ by considering only those datacubes which satisfy a given set of integrity constraints on the number of non-null elements.

Let us now define the random variables for the estimation of the count and the sum query.

Let the query $count(M[\mathbf{i}..\mathbf{j}])$ and $sum(M[\mathbf{i}..\mathbf{j}])$ be given and let $LB_{=0}$ and $LB_{>0}$ be two integrity constraints that are satisfied by M. We shall estimate the two queries with the two random query variables $count(\tilde{M}[\mathbf{i}..\mathbf{j}])$ and $sum(\tilde{M}[\mathbf{i}..\mathbf{j}])$, respectively, in the following two cases:

1. for $\tilde{M}$ extracted from the population $\sigma_{LB_{=0}}(M^{-1}_{cs,F}) = \{M' |\ M' \in M^{-1}_{cs,F}$ and M' satisfies $LB_{=0}\}$; thus we estimate the number and the sum of the non-null elements in $M[\mathbf{i}..\mathbf{j}]$ by considering the population of all datacubes having both the same sum and the same number of non-nulls in each block as M and satisfying the lower bound constraint enforced by the function $LB_{=0}$ on the number of null elements occurring in each range;
2. for $\tilde{M}$ extracted from the population $\sigma_{LB_{=0},LB_{>0}}(M^{-1}_{cs,F}) = \{M' |\ M' \in M^{-1}_{cs,F}$ and M' satisfies $LB_{=0}$ and $LB_{>0}\}$; thus we estimate the number and the sum of the non-null elements in $M[\mathbf{i}..\mathbf{j}]$ by restricting the population of the previous case to those datacubes which also satisfy the lower bound constraint enforced by the function $LB_{>0}$ on the number of non-null elements occurring in each range.

We observe that Case 1 can be derived from the more general Case 2 but, for the sake of presentation, we first present the simpler case and then we move to the general case.

Once the datacube population for a random variable $query(\tilde{M}[\mathbf{i}..\mathbf{j}])$ (where *query* stands for *count* or *sum*) is fixed, we have to determine its probability distribution and then its mean and variance — recall that both mean and variance are defined by the operator E. Concerning the mean, due to the linearity of E we have:

$$E(query(\tilde{M}[\mathbf{i}..\mathbf{j}]) = \sum_{B_{\mathbf{q}} \in TB_F(\mathbf{i}..\mathbf{j})} M_{query,F}[\mathbf{q}] + \sum_{B_{\mathbf{k}} \in PB_F(\mathbf{i}..\mathbf{j})} E(query(\tilde{M}[\mathbf{i_k}..\mathbf{j_k}]))$$

where

1. $TB_F(\mathbf{i..j})$ returns the set of blocks $B_\mathbf{q}$ that are totally contained in the range $[\mathbf{i..j}]$, i.e., both $\mathbf{i} \leq F^-(\mathbf{q})$ and $F^+(\mathbf{q}) \leq \mathbf{j}$,
2. $PB_F(\mathbf{i..j})$ returns the set of blocks $B_\mathbf{k}$ that are partially inside the range, i.e., $B_\mathbf{k} \notin TB_F(\mathbf{i..j})$ and either $\mathbf{i} \leq F^-(\mathbf{k}) \leq \mathbf{j}$ or $\mathbf{i} \leq F^+(\mathbf{k}) \leq \mathbf{j}$, and
3. for each $B_\mathbf{k} \in PB_F(\mathbf{i..j})$, $\mathbf{i_k}$ and $\mathbf{j_k}$ are the boundaries of the portion of the block $B_\mathbf{k}$ which overlaps the range $[\mathbf{i..j}]$, i.e., $[\mathbf{i_k..j_k}] = [\mathbf{i..j}] \cap [F^-(\mathbf{k})..F^+(\mathbf{k})]$.

For example, for the datacube in Figure 1(a), given $\mathbf{i} = <4,3>$ and $\mathbf{j} = <8,6>$, the block $B_{<2,2>}$ is totally contained in the range, the blocks $B_{<2,1>}$, $B_{<3,1>}$, $B_{<3,2>}$ are partially contained in the range (with boundaries $[4..7, 3..4]$, $[8..8, 3..4]$ and $[8..8, 5..6]$, respectively), and the blocks $B_{<1,1>}$, $B_{<1,2>}$ are outside the range.

Concerning the variance, we assume statistical independence between the measure values of different blocks so that its value is determined by summing the variances of all partially overlapped blocks, thus introducing no covariance.

$$\sigma^2(query(\tilde{M}[\mathbf{i..j}]) = \sum_{B_\mathbf{k} \in PB_F(\mathbf{i..j})} \sigma^2(query(\tilde{M}[\mathbf{i}_k..\mathbf{j}_k])).$$

It turns out that we only need to study the estimation of a query ranging on one partial block as all other cases can be easily re-composed from this basic case. Therefore, from now on we assume that the query range $[\mathbf{i..j}]$ is strictly inside one single block, say the block $B_\mathbf{k}$, i.e., $F^-(\mathbf{k}) \leq \mathbf{i} \leq \mathbf{j} \leq F^+(\mathbf{k})$. We use the following notations and assumptions:

1. b, $b > 1$, is the size of $B_\mathbf{k}$, thus b is the number of elements in $B_\mathbf{k}$;
2. $b_{\mathbf{i..j}}$, $1 \leq b_{\mathbf{i..j}} < b$, is the size of $[\mathbf{i..j}]$, that is the number of elements in the range;
3. $t = M_{count,F}[\mathbf{k}]$, $1 \leq t \leq b$, is the number of non-null elements in $B_\mathbf{k}$;
4. $s = M_{sum,F}[\mathbf{k}]$, $s \geq max(1,t)$, is the sum of the elements in $B_\mathbf{k}$;
5. $t^U_{\mathbf{i..j}} = b_{\mathbf{i..j}} - LB_{=0}([\mathbf{i..j}])$ and $t^L_{\mathbf{i..j}} = LB_{>0}([\mathbf{i..j}])$ are respectively an upper bound and a lower bound on the number of non-null elements in the range $[\mathbf{i..j}]$;
6. $t^U_{\widetilde{\mathbf{i..j}}} = b_{\widetilde{\mathbf{i..j}}} - LB_{=0}([\widetilde{\mathbf{i..j}}])$ and $t^L_{\widetilde{\mathbf{i..j}}} = LB_{>0}([\widetilde{\mathbf{i..j}}])$ are respectively an upper bound and a lower bound on the number of non-null elements in the block B_k outside the range $[\mathbf{i..j}]$;
7. $t^U = t^U_{\mathbf{i..j}} + t^U_{\widetilde{\mathbf{i..j}}} = b - LB_{=0}([\mathbf{i..j}]) - LB_{=0}([\widetilde{\mathbf{i..j}}])$ and $t^L = t^L_{\mathbf{i..j}} + t^L_{\widetilde{\mathbf{i..j}}} = LB_{>0}([\mathbf{i..j}]) + LB_{>0}([\widetilde{\mathbf{i..j}}])$, where $[\widetilde{\mathbf{i..j}}]$ denotes the set of elements that are in $B_\mathbf{k}$ but not in the range $[\mathbf{i..j}]$; t^U and t^L are an upper bound and a lower bound on the number of non-null elements in $B_\mathbf{k}$.

Observe that the functions $LB_{=0}$ and $LB_{>0}$ are computed for the ranges $[\mathbf{i..j}]$ and $[\widetilde{\mathbf{i..j}}]$ but not for the whole block B_k. Indeed, t^L and t^U do not in general coincide with $LB_{>0}([F^-(\mathbf{k})..F^+(\mathbf{k})])$ and $b - LB_{=0}([F^-(\mathbf{k})..F^+(\mathbf{k})])$, respectively, as the latter ones may be stricter bounds. For instance, suppose that the block stores the bimonthly sales of a store and we want to estimate the sales in the first month. The integrity constraints say that the store closes 4

days every month and an additional day every two months. So $t^U = 60 - 4 = 56$ and not 55. Thus the additional day is not taken into account but this does not affect at all the accuracy of the estimation: indeed we have available the actual number of opened days for the block of two months.

4 Case 1: Range Query Estimation using Upper Bounds on the Number of Non-Null Elements

In this section we only consider upper bounds on the number of non-null elements in the ranges $[\mathbf{i..j}]$ and $[\mathbf{i\widetilde{.}j}]$ which are derived by the function $LB_{=0}$. We define the random variables $count(\tilde{M}[\mathbf{i..j}])$ and $sum(\tilde{M}[\mathbf{i..j}])$ by extracting $\tilde{M}$ from the population $\sigma_{LB_{=0}}(M^{-1}_{cs,F})$ of all datacubes having both the same sum and the same number of non-nulls in each block as M and satisfying the upper bound constraints on the number of elements in each range. We assume that both $t^L_{\mathbf{i..j}}$ and t^L are equal to zero, i.e., both $LB_{>0}([\mathbf{i..j}]) = 0$ and $LB_{>0}([\mathbf{i\widetilde{.}j}]) = 0$.

Theorem 1. *Let $C_1([\mathbf{i..j}]) = count(\tilde{M}[\mathbf{i..j}])$ and $S_1([\mathbf{i..j}]) = sum(\tilde{M}[\mathbf{i..j}])$ be two integer random variables ranging from 0 to t and from 0 to s, respectively, defined by taking $\tilde{M}$ in the datacube population $\sigma_{LB_{=0}}(M^{-1}_{cs,F})$. If $t^L_{\mathbf{i..j}} = t^L = 0$ then for each $t_{\mathbf{i..j}}$ and $s_{\mathbf{i..j}}$, $0 \leq t_{\mathbf{i..j}} \leq t^U_{\mathbf{i..j}}$ and $0 \leq s_{\mathbf{i..j}} \leq s$, the joint probability distribution $P(C_1([\mathbf{i..j}]) = t_{\mathbf{i..j}}, S_1([\mathbf{i..j}]) = s_{\mathbf{i..j}})$ is equal to:*

$$P(C_1([\mathbf{i..j}]) = t_{\mathbf{i..j}}, S_1([\mathbf{i..j}]) = s_{\mathbf{i..j}}) = \frac{Q(t^U_{\mathbf{i..j}}, t_{\mathbf{i..j}}, s_{\mathbf{i..j}}) \cdot Q(t^U_{\mathbf{i\widetilde{.}j}}, t_{\mathbf{i\widetilde{.}j}}, s_{\mathbf{i\widetilde{.}j}})}{Q(t^U, t, s)}$$

where $t_{\mathbf{i\widetilde{.}j}} = t - t_{\mathbf{i..j}}$, $s_{\mathbf{i\widetilde{.}j}} = s - s_{\mathbf{i..j}}$, and

$$Q(\overline{t}_u, \overline{t}, \overline{s}) = \begin{cases} 0 & \text{if } (\overline{t} = 0 \wedge \overline{s} > 0) \vee (\overline{t} > 0 \wedge \overline{s} < \overline{t}) \vee \overline{t} > \overline{t}_u \\ 1 & \text{if } \overline{t} = 0 \wedge \overline{s} = 0 \\ \binom{\overline{t}_u}{\overline{t}} \cdot \binom{\overline{s}-1}{\overline{s}-\overline{t}} & \text{otherwise.} \end{cases}$$

Proof. (Sketch) The probability distribution does not change if we reduce the size of the range and of the block by removing certain null elements. Therefore, the size of the block $B_{\mathbf{k}}$ is assumed to be t^U and, then, the size of the query query becomes $t^U_{\mathbf{i..j}}$. We can now see the block as a vector, say V, of t^U elements with values in $[1..s]$ such that their total sum is s and the number of non null elements is t. We divide V into two subvectors V' and V'' such that V' consists of the first $t^U_{\mathbf{i..j}}$ elements and V'' of the last $t^U_{\mathbf{i\widetilde{.}j}} = t^U - t^U_{\mathbf{i..j}}$ ones. The probability of the event $(C_1([\mathbf{i..j}]) = t_{\mathbf{i..j}} \wedge S_1([\mathbf{i..j}]) = s_{\mathbf{i..j}})$ is then equal to the probability that V' contains $t_{\mathbf{i..j}}$ non-null elements whose sum is $s_{\mathbf{i..j}}$. Let denote by $\bar{P}$ this probability. Observe that the event implies that V'' contains $t - t_{\mathbf{i..j}}$ non null-elements whose sum is $s - s_{\mathbf{i..j}}$. It is then easy to see that $\bar{P}$ is equal to

$$\frac{Q(t^U_{\mathbf{i..j}}, t_{\mathbf{i..j}}, s_{\mathbf{i..j}}) \cdot Q(t^U - t^U_{\mathbf{i..j}}, t - t_{\mathbf{i..j}}, s - s_{\mathbf{i..j}})}{Q(t^U, t, s)}$$

where $Q(\overline{t}_u, \overline{t}, \overline{s})$ is the number of possible configurations for a vector of size $\overline{t}_u$ containing exactly $\overline{t}$ non-null elements with total sum $\overline{s}$.

$Q(\overline{t}_u, \overline{t}, \overline{s})$ can be determined by considering all possible ways of distributing the sum $\overline{s}$ into $\overline{t}$ non-fixed positions by assigning to each of such elements a value from 1 to s. If we fix the positions of the $\overline{t}$ non-null elements,we obtain that the number of such configurations is:

$$m(\overline{t}, \overline{s}) = \binom{\overline{t} + (\overline{s} - \overline{t}) - 1}{\overline{s} - \overline{t}} = \binom{\overline{s} - 1}{\overline{s} - \overline{t}}.$$

As the positions for the $\overline{t}$ non-null elements are not fixed, we have to multiply $m(\overline{t}, \overline{s})$ by the number of all possible dispositions of $\overline{t}$ non-nulls over xd positions, that is

$$n(\overline{t}_u, \overline{t}) = \binom{\overline{t}_u}{\overline{t}}.$$

Hence, $Q(\overline{t}_u, \overline{t}, \overline{s}) = n(\overline{t}_u, \overline{t}) \cdot m(\overline{t}, \overline{s})$. □

Mean and variance of the random variable $C_1([\mathbf{i}..\mathbf{j}])$ are presented in the next proposition.

Proposition 1. *Let $C_1([\mathbf{i}..\mathbf{j}])$ be the random variables defined in Theorem 1. Then, mean and variance are, respectively:*

$$E(C_1([\mathbf{i}..\mathbf{j}])) = \frac{t^U_{\mathbf{i}..\mathbf{j}}}{t^U} \cdot t$$

$$\sigma^2(C_1([\mathbf{i}..\mathbf{j}])) = t \cdot (t^U - t) \cdot t^U_{\mathbf{i},\mathbf{j}} \cdot \frac{t^U_{\widetilde{\mathbf{i}..\mathbf{j}}}}{(t^U)^2 \cdot (t^U - 1)}$$

Proof. (Sketch) Consider the vector V, V' and V'' defined in the proof of Theorem 1. The event $(C_1([\mathbf{i}..\mathbf{j}]) = t_{\mathbf{i}..\mathbf{j}})$ is equivalent to the event that V' contains exactly $t_{\mathbf{i}..\mathbf{j}}$ non-null elements. Observe that the probabilty that an element is not null is equal to t/t^U. Hence $(C_1([\mathbf{i}..\mathbf{j}]) = t_{\mathbf{i}..\mathbf{j}})$ is in turn equivalent to the event of extracting $t_{\mathbf{i}..\mathbf{j}}$ non-nulls from V in $t^U_{\mathbf{i}..\mathbf{j}}$ trials. This probability is described by the well-known hypergeometric distribution [7]. □

Now we determine mean and variance of the random variable $S_1([\mathbf{i}..\mathbf{j}])$.

Theorem 2. *Consider the random variable $S_1([\mathbf{i}..\mathbf{j}])$ defined in Theorem 1. Then, mean and variance of $S_1([\mathbf{i}..\mathbf{j}])$ are, respectively:*

$$E(S_1([\mathbf{i}..\mathbf{j}])) = \frac{t^U_{\mathbf{i}..\mathbf{j}}}{t^U} \cdot s$$

$$\sigma^2(S_1([\mathbf{i}..\mathbf{j}])) = \frac{s \cdot t^U_{\mathbf{i},\mathbf{j}} \cdot t^U_{\widetilde{\mathbf{i}..\mathbf{j}}}}{t^{U\,2} \cdot (t^U - 1) \cdot (t + 1)} \cdot [t^U \cdot (2 \cdot s - t + 1) - s \cdot (t + 1)].$$

Proof. (Sketch) Consider the vector V, V' and V'' defined in the proof of Theorem 1. The event $(S_1([\mathbf{i..j}]) = s_{\mathbf{i..j}})$ is equivalent to the following event: the sum of all elements in V' is $s_{\mathbf{i..j}}$. From $s = \sum_{1 \leq i \leq t^U} V[i]$, we derive $s = \sum_{1 \leq i \leq t^U} E(V[i])$ by linearity of the operator E. Further, the mean of random variable $V[i]$ is equal to the mean of the random variable $V[j]$, for any i, j, $1 \leq i, j \leq t^U$. Indeed, for symmetry, the probability that an element of V assumes a given value is independent on the position of this element inside the vector. Let denote by m this mean. From the above formula for s it then follows that $m \cdot t^U = s$, thus $t = s/t^U$. Consider now the vector V'. Let S' be the random variable representing the sum of all elements of V'. Then $E(S') = t^U_{\mathbf{i..j}} \cdot m$. Hence, $E(S') = t^U_{\mathbf{i..j}} \cdot s/t^U$.

The variance can be obtained using its definition. To this end, we first need to determine the probability distribution of $S_1([\mathbf{i..j}])$ from the joint probability distribution obtained in Theorem 1. The detailed proof is rather elaborated and, for the sake of presentation, is included in the appendix. □

Note that the mean of the random variable $S_1([\mathbf{i..j}])$ representing the sum query does not depend on the number t of non-null elements in the block $B_{\mathbf{k}}$. On the other hand, the knowledge about certain null elements derived by the function $LB_{=0}$ does influence the value of the sum. Indeed, the mean depends both on the size of the query range w.r.t. the size of the block and on the number of the nulls that are already known to be in the range and in the complementary part of the block.

5 Case 2: Range Query Estimation using both Lower Bounds and Upper Bounds on on the Number of Non-Null Elements

We are now ready to perform the estimation of range queries in the general case where the datacube population is the set $\sigma_{LB_{=0},LB_{>0}}(M^{-1}_{cs,F})$ of all datacubes having the same aggregate data (count and sum) as M and satisfying both constraints: the lower bound on the number of null elements occurring in each range and the lower bound on the number of non-null elements.

Theorem 3. *Let $C_2([\mathbf{i..j}]) = count(\tilde{M}[\mathbf{i..j}])$ and $S_2([\mathbf{i..j}]) = sum(\tilde{M}[\mathbf{i..j}])$ be two integer random variable ranging from 0 to t and from 0 to s, respectively, defined by taking $\tilde{M}$ in the datacube population $\sigma_{LB_{=0},LB_{>0}}(M^{-1}_{cs,F})$. Then, for each $t_{\mathbf{i..j}}$ and $s_{\mathbf{i..j}}$, $t^L_{\mathbf{i..j}} \leq t_{\mathbf{i..j}} \leq t^U_{\mathbf{i..j}}$ and $0 \leq s_{\mathbf{i..j}} \leq s$, the joint probability distribution $P(C_2([\mathbf{i..j}]) = t_{\mathbf{i..j}}, S_2([\mathbf{i..j}]) = s_{\mathbf{i..j}})$ is equal to:*

$$P(C_2([\mathbf{i..j}]) = t_{\mathbf{i..j}}, S_2([\mathbf{i..j}]) = s_{\mathbf{i..j}}) = \frac{N(t^U_{\mathbf{i..j}}, t_{\mathbf{i..j}}, s_{\mathbf{i..j}}, t^L_{\mathbf{i..j}}) \cdot N(t^U_{\widetilde{\mathbf{i..j}}}, t_{\widetilde{\mathbf{i..j}}}, s_{\widetilde{\mathbf{i..j}}}, t^L_{\widetilde{\mathbf{i..j}}})}{N(t^U, t, s, t^L)}$$

where $t_{\mathbf{i..j}}^{\sim} = t - t_{\mathbf{i..j}}$, $s_{\mathbf{i..j}}^{\sim} = s - s_{\mathbf{i..j}}$, *and*

$$N(\overline{t}_u, \overline{t}, \overline{s}, \overline{t}_l) = \begin{cases} 0 & \text{if } \overline{t} > \overline{t}_u \vee \overline{t} > \overline{s} \vee (\overline{t} = 0 \wedge \overline{s} > 0) \\ 1 & \text{if } \overline{t} = 0 \wedge \overline{s} = 0 \\ \binom{\overline{t}_u - \overline{t}_l}{\overline{t} - \overline{t}_l} \cdot \binom{\overline{s} - 1}{\overline{s} - \overline{t}} & \text{otherwise} \end{cases}$$

Proposition 2. *Consider the random variable* $C_2([\mathbf{i..j}])$ *of Theorem 3. Then, mean and variance are:*

$$E(C_2([\mathbf{i..j}])) = t_{\mathbf{i..j}}^L + \frac{t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L}{t^U - t^L} \cdot (t - t^L)$$

$$\sigma^2(C_2([\mathbf{i..j}])) = \frac{t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L}{t^U - t^L} \cdot (t - t^L) \cdot \frac{[(t^U - t^L) - (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L)] \cdot (t^U - t)}{(t^U - t^L) \cdot (t^U - t^L - 1)}$$

Theorem 4. *Consider the random variable* $S_2([\mathbf{i..j}])$ *of Theorem 3. Then, mean and variance are:*

$$E(S_2([\mathbf{i..j}])) = t_{\mathbf{i..j}}^L \cdot \frac{s}{t} + (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L) \cdot \frac{s}{t} \cdot \frac{t - t^L}{t^U - t^L}.$$

$$\begin{aligned} \sigma^2(S_2([\mathbf{i..j}])) = \alpha \cdot (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L) \cdot \tfrac{t - t^L}{t^U - t^L} \cdot \left[1 + (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L - 1) \cdot \tfrac{t - t^L - 1}{t^U - t^L - 1}\right] + \\ (\beta + 2 \cdot \alpha \cdot t_{\mathbf{i..j}}^L) \cdot (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L) \cdot \tfrac{t - t^L}{t^U - t^L} + (\alpha \cdot {t_{\mathbf{i..j}}^L}^2 + \beta \cdot t_{\mathbf{i..j}}^L) - \gamma^2 \end{aligned}$$

where:

$$\alpha = \frac{s \cdot (s+1)}{t \cdot (t+1)}, \beta = \frac{s \cdot (s-t)}{t \cdot (t+1)} \text{ and } \gamma = t_{\mathbf{i..j}}^L \cdot \frac{s}{t} + (t_{\mathbf{i..j}}^U - t_{\mathbf{i..j}}^L) \cdot \frac{s}{t} \cdot \frac{t - t^L}{t^U - t^L}.$$

Note that, unlike the case 1, the mean $S_2([\mathbf{i..j}])$ of the random variable representing the sum query depends on the number t of non null elements occurring in the block $B_{\mathbf{k}}$. Indeed, in this case, the information encoded in the function $LB_{>0}$ actually invalidates the symmetry condition about the aggregate information used for the estimation, on which the independence of the mean from t is based. This happens since $LB_{>0}$ returns a number of certain non-null elements, thus giving a positive contribution both to the count and the sum query. Thus, such positions cannot be eliminated in order to re-formulate the query into a new query applied on a block with indistinguishable positions as it happens for the Case 1.

Also in this case, the mean is not in general a linear function respect to the size of the query, since it depends both on $t_{\mathbf{i..j}}^U$ (and then on $LB_{=0}([\mathbf{i..j}])$) and on $t_{\mathbf{i..j}}^L$ (and then on $LB_{>0}([\mathbf{i..j}])$). Thus, once again, the estimation depends on the actual distribution of the values inside the block and not only on the aggregate information concerning count and sum of the block.

6 Estimation of Range Queries on Histograms

Histograms are mono-dimensional compressed datacube that are used to summarize the frequency distribution of an attribute of a database relation for the estimation of query result sizes [14–16]. The estimation is made using aggregate data such as the number t of non-null values in each block (bucket in the histogram terminology) B_k, the total frequency sum s in B_k and the boundaries of B_k. A crucial point for providing good estimations is the way the frequency distributions for original values are partitioned into buckets. Here we assume that the buckets have been already arranged using any of the known techniques and we therefore focus on the problem of estimating the frequency distribution inside a bucket.

The most common approach is based on the *continuous value assumption* [17]: the sum of frequencies in a range of a bucket is estimated by linear interpolation. It thus corresponds to equally distributing the overall sum of frequencies of the bucket to all attribute values occurring in it. This result can be derived from Theorem 2 by assuming that there are no integrity constraints on the number of null and non-null elements.

Corollary 1. *Let $B_{\mathbf{k}}$ be a block of a histogram and let $S_3([\mathbf{i}..\mathbf{j}]) = sum(\tilde{M}[\mathbf{i}..\mathbf{j}])$ be an integer random variable ranging from 0 to s, defined by taking $\tilde{M}$ in the datacube population $M^{-1}_{cs,F}$. Then mean and variance of $S_3([\mathbf{i}..\mathbf{j}])$ are, respectively:*

$$E(S_3([\mathbf{i}..\mathbf{j}])) = \frac{b_{\mathbf{i}..\mathbf{j}}}{b} \cdot s$$

$$\sigma^2(S_3([\mathbf{i}..\mathbf{j}])) = \frac{s \cdot b_{\mathbf{i}..\mathbf{j}} \cdot (b - b_{\mathbf{i}..\mathbf{j}})}{b^2 \cdot (b-1) \cdot (t+1)} \cdot [b \cdot (2 \cdot s - t + 1) - s \cdot (t+1)].$$

Thus our approach gives a model to explain the linear interpolation and, besides, allows to evaluate the error of the estimation, thus exploiting the knowledge about the number t of non-nulls in a block — instead t is not mentioned in the computation of the mean.

We now recall that the classical definition of histogram requires that both lowest and highest elements (or at least one of them) of any block are not null (i.e., they are attribute values occurring in the relation). A block for which the extreme elements are not null are called *2-biased*; if only the lowest (or the highest) element is not null then the block is called *1-biased*.

So far linear interpolation is also used for biased blocks thus producing a wrong estimation — it is the case to say a "biased" estimation. We next show the correct formulas that are derived from Theorem 4.

Corollary 2. *Let $B_{\mathbf{k}}$ be a block of a histogram and let $S_4([\mathbf{i}..\mathbf{j}]) = sum(\tilde{M}[\mathbf{i}..\mathbf{j}])$ be an integer random variable ranging from 0 to s, defined by taking $\tilde{M}$ in the datacube population $\sigma_{LB_{>0}}(M^{-1}_{cs,F})$. Then*

1. *if the block $B_{\mathbf{k}}$ is 1-biased and* $\mathbf{i}$ *is the lowest element of the block then mean and variance of* $S_4([\mathbf{i..j}])$ *are, respectively:*

$$E(S_4([\mathbf{i..j}])) = \frac{s}{t} + (b_{\mathbf{i..j}} - 1) \cdot \frac{s}{t} \cdot \frac{t-1}{b-1},$$

$$\sigma^2(S_4([\mathbf{i..j}])) = \alpha \cdot (b_{\mathbf{i..j}} - 1) \cdot \tfrac{t-1}{b-1} \cdot \left[1 + (b_{\mathbf{i..j}} - 2) \cdot \tfrac{t-2}{b-2}\right] + (\beta + 2 \cdot \alpha) \cdot (b_{\mathbf{i..j}} - 1) \cdot \tfrac{t-1}{b-1} + (\alpha + \beta) - E(S_4([\mathbf{i..j}]))^2$$

2. *if the block $B_{\mathbf{k}}$ is 1-biased and* $\mathbf{i}$ *is not the lowest element of the block then mean and variance of* $S_4([\mathbf{i..j}])$ *are, respectively:*

$$E(S_4([\mathbf{i..j}])) = b_{\mathbf{i..j}} \cdot \frac{s}{t} \cdot \frac{t-1}{b-1},$$

$$\sigma^2(S_4([\mathbf{i..j}])) = \alpha \cdot b_{\mathbf{i..j}} \cdot \frac{t-1}{b-1} \cdot \left[1 + (b_{\mathbf{i..j}} - 1) \cdot \frac{t-2}{b-2}\right] + \beta \cdot b_{\mathbf{i..j}} \cdot \frac{t-1}{b-1} - E(S_4([\mathbf{i..j}]))^2$$

3. *if the block $B_{\mathbf{k}}$ is 2-biased and either* $\mathbf{i}$ *or* $\mathbf{j}$ *is an extreme element of the block then mean and variance of* $S_4([\mathbf{i..j}])$ *are, respectively:*

$$E(S_4([\mathbf{i..j}])) = \frac{s}{t} + (b_{\mathbf{i..j}} - 1) \cdot \frac{s}{t} \cdot \frac{t-2}{b-2},$$

$$\sigma^2(S_4([\mathbf{i..j}])) = \alpha \cdot (b_{\mathbf{i..j}} - 1) \cdot \tfrac{t-2}{b-2} \cdot \left[1 + (b_{\mathbf{i..j}} - 2) \cdot \tfrac{t-3}{b-3}\right] + (\beta + 2 \cdot \alpha) \cdot (b_{\mathbf{i..j}} - 1) \cdot \tfrac{t-2}{b-2} + (\alpha + \beta) - E(S_4([\mathbf{i..j}]))^2$$

4. *if the block $B_{\mathbf{k}}$ is 2-biased and neither* $\mathbf{i}$ *nor* $\mathbf{j}$ *is an extreme element of the block then mean and variance of* $S_4([\mathbf{i..j}])$ *are, respectively:*

$$E(S_4([\mathbf{i..j}])) = b_{\mathbf{i..j}} \cdot \frac{s}{t} \cdot \frac{t-2}{b-2},$$

$$\sigma^2(S_4([\mathbf{i..j}])) = \alpha \cdot b_{\mathbf{i..j}} \cdot \frac{t-2}{b-2} \cdot \left[1 + (b_{\mathbf{i..j}} - 1) \cdot \frac{t-3}{b-3}\right] + \beta \cdot b_{\mathbf{i..j}} \cdot \frac{t-2}{b-2} - E(S_4([\mathbf{i..j}]))^2$$

where:

$$\alpha = \frac{s \cdot (s+1)}{t \cdot (t+1)} \text{ and } \beta = \frac{s \cdot (s-t)}{t \cdot (t+1)}.$$

The above formulas have been used in [4] to replace the continuous value assumption inside one of the most efficient methods for histogram representation (the maxdiff method [16]) and have produced some meaningful improvements in the performance of the method.

In [16, 15], another method for estimating frequency sum inside a block is proposed, based on the *uniform spread assumption*: the t non-null attribute values in each bucket are assumed to be located at equal distance from each other and the overall frequency sum is therefore equally distributed among them. This method does not give a correct estimation unless we assume that nun-nulls are scattered on the block in some particular, unrealistic way. Our approach gives instead an unbiased estimation.

References

1. S. Acharya, P.B. Gibbons, Poosala, S. Ramaswamy. Join Synopses for Approximate Query Answering, In *Proc. of SIGMOD International Conference On Management Of Data* June 1999.
2. Barbara, D., DuMouchel, W., Faloutsos, C., Haas, P.J., Hellerstein, J.M., Ionnidis, Y., Jagadish, H.V., Johnson, T., Ng, R., Poosala, V., Ross, K.A., Sevcik, K.C., The New Jersey data reduction report, *Bulletin of the Technical Committee on Data Engineering 20*, 4, 3-45, 1997.
3. Buccafurri, F., Rosaci, D., Sacca', D., Compressed datacubes for fast OLAP applications, *DaWaK 1999*, Florence, 65-77.
4. Buccafurri, F., Pontieri, L., Rosaci, D., Sacca', D., Improving Range Query Estimation on Histograms, unpublished manuscript, 2000.
5. Chaudhuri, S., Dayal, U., An Overview of Data Warehousing and OLAP Technology, *ACM SIGMOD Record 26(1)*, March 1997.
6. C. Faloutsos, H. V. Jagadish, N. D. Sidiripoulos. Recovering Information from Summary Data. In *Proceedings of the 1997 VLDB Very Large Data Bases Conference*, Athens, 1997
7. W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 1968.
8. P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998
9. P.B.Gibbons, Y.Matias, V.Poosala. AQUA Project White Paper, At *http://www.bell-labs.com/user/pbgibbons/papers*, 1997.
10. P.B.Gibbons, Y.Matias, V.Poosala. Fast incremental maintenance of approximate histograms, *Proc. of the 23rd VLDB Conf.*, 466-475, August 1997.
11. Gray, J., Bosworth, A., Layman, A., Pirahesh, H., Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, *Proc. of the ICDE 1996*, pp. 152-159
12. J. M. Hellerstein, P. J. Haas, H. J. Wang. Online Aggregation. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data*, pages 171-182, 1997
13. Harinarayan, V., Rajaraman, A., Ullman, J. D., Implementing Data Cubes Efficiently, *Proc. of the ACM SIGMOD 1996*, pp. 205-216
14. Y. Ioannidis, V. Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 233-244, 1995
15. V. Poosala. *Histogram-based Estimation Techniques in Database Systems*. PhD dissertation, University of Wisconsin-Madison, 1997
16. V. Poosala, Y. E. Ioannidis, P. J. Haas, E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 294-305, 1996
17. P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. T. Price. Access path selection in a relational database management system. In *Proc. of ACM SIGMOD Internatinal Conference*, pages 23-24, 1979
18. J. S. Vitter, M. Wang, B. Iyer. Data Cube Approximation and Histograms via Wavelets. In *Proceedings of the 1998 CIKM International Conference on Information and Knowledge Management*, Washington, 1998