Proceedings of the Federated Conference on Computer Science and Information Systems pp. 9–15
DOI: 10.15439/2016F543
ACSIS, Vol. 8. ISSN 2300-5963

Quality of Histograms As Indicator Of Approximate Query Quality

Agnieszka Chądzyńska-Krasowska
Polish-Japanese Academy of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
Email: honzik@pjwstk.edu.pl
Infobright Inc.
ul. Krzywickiego 34/219, 02-078 Warsaw, Poland
Email: agnieszka.chadzynska-krasowska@infobright.com

Marcin Kowalski
Infobright Inc.
ul. Krzywickiego 34/219, 02-078 Warsaw, Poland,
Email: marcin.kowalski@infobright.com

***Abstract*—We consider concept of *approximate query* in RDBMS i.e. query that returns results which may differ from common (exact) query results in a way but its evaluation requires less resources. In the work we focus mostly on time and storage space aspects. We follow one of the state-of-the-art trends using synopses of data as the input of approximate query evaluation. We propose some measures of approximate query results quality. Basing on them we present steps of adaptive elaboration of synopses quality measure that should be mutually corresponding.**

***Index Terms*—Approximate Query, Quality Measures, Histograms.**

I. Introduction

APPROXIMATE query concept emerged as a tool of coping with continuous growth of data volume gathered in databases. Disposing limited budget of resources like time, storage space, computing power etc. database user wanted to gain information from the data quickly but accepting fact that achieved results may not be crisp. Two main classes of solutions are present in the literature: data sampling techniques [1], [2] and data synopses calculation [3], [4]. The former one utilizes statistical apparatus for choosing representative sample of the data, considerably smaller than whole data set, and using it to estimate results of query for whole data. The latter is based on concepts of data synopses which are data descriptions built once during load, stored and exclusively used during query evaluation. Some compact comparison between both approaches is contained in e.g. [5].

In the world of RDBMSs synopses are considered as the descriptions of e.g. columns value data sets. We may consider both one or multicolumns descriptions. The most common and well-examined types of data synopses described in the literature are histograms. In standard approaches, histograms are built per whole column (or more columns) and inference based on histograms is done for whole relation.

Our approach differs from this while we utilized some elements of *granular computing*[1] in the query evaluation process. Despite our approach may be treated as an example of the synopses calculus trend, we create synopses and inference on their basis not on whole relation level but on their parts. This may require some additional operations during query evaluation like compound partial results of evaluated query from each data packs. On the other hand it enables reflecting potential changes or differences in columns value sets in time and avoiding complex operations on synopses when e.g. consecutive loads into relations are considered.

Other consequence of this fact is that the size (in bytes) of generated synopses must be of orders of magnitude smaller than compressed data themselves and synopses build in standard approaches for whole column. For effectiveness sake and in order to easily control storage we assume also existing size budget for single synopsis. That means in particular that for most cases we cannot afford storing e.g. exact histogram of value set from part of the table (i.e. histogram in which all intervals are single-valued). The idea of utilizing granularity concepts may go further in building hierarchical structures of synopses e.g. for subsets of row/data packs.

The presented analysis concerns measures of histogram quality. We will say that the histogram is of good quality when applying used methodology led to achieve good approximation of query results. That raises immediately question of quality of the latter one. In the article we introduce measures of quantifying approximate query results and their selected aspects. The main purpose of the work remains though developing and analyzing measures of histogram quality that will directly translate to proposed quality measures of the query results.

Section II describes basics of the Infobright RDBMS engine that we had chosen for experiments, explains terminology we use in the work and presents methodology used in experiments framework. We discuss problem of measuring approximate query results in Section III and propose some formulas for it. In Section IV we describe standard methods of 1dim histogram generation and classic quality measures for them. As the experiments result presented in latter section run for standard approaches were not satisfactory, we propose in Section V some modifications of them.

[1]https://en.wikipedia.org/wiki/Granular_computing

II. Basic Definitions and Methodology

A. Definitions

We consider *1–dimensional histograms* as a collection of *bars* (*buckets*), each of which then consists of: interval (determined by 2 numbers - interval's begin and end) and *frequency* which in our work is cardinal (but finite) number. If single histogram bars' intervals are disjoint, frequencies should be interpreted as number of elements which occur in the interval. Exemplary 1dim histograms are in Figures 3 – 5 . We may easily extend this definition to *multidimensional histograms* changing intervals to *cubes* however we do not consider multidimensional case in the article so we omit it here. Histograms might be generated using many algorithms. Some of them are presented in Section IV. In the work we may refer to information contained in synopses as to *rough data* or *granuled data*.

B. Infobright Basics

In the considered RDBMS engine [6], [7], during the data load process, the sets of relation tuples are split into chunks of equal and fixed cardinality (apart from the last chunk perhaps) which form *row packs*. Sequence of values from row pack for single column forms *data pack* for this column. As *data pack range* we mark interval from minimal to maximal values of data pack. In actual implementation of the database engine chunk size is equal to 2^{16}. Data packs are then compressed separately and stored. For each of formed data pack there is additionally built and stored compact information (data synopsis) of values from data pack. Such synopsis may be considered as the information granule of the data pack. Prepared data synopses are loaded into memory during query evaluation and used in several ways e.g. to filter out irrelevant data (e.g. to avoid costly I/O operations for corresponding data packs).

As pointed out, in the described engine, both compressed data packs and synopses are stored. In some of our previous works, we investigated how to further decrease the amounts of accessed data packs and focus more on synopses-based computations in order to accelerate computations, on the cost of possible inexactness of query results [8], [9]. However, in the case of the new Infobright's engine dedicated to approximate queries only synopses are stored, while exact data are accessible only during their load (when synopses are being generated) and then forgotten.[2] This is also the assumption that we follow in this paper. Another aspect that distinguishes our approach from most of standard methods is the fact that our data synopses are built for each data pack – not for each column only.

C. Experiments Framework

As there is no room for describing methods of using synopses in internal operations of the query engine and, on the other hand, this is not the main purpose of this article, we decide to emulate behaviour of the engine designed for approximate queries. Generally speaking, histogram reflects probability distribution and therefore might be the input of random value generator. To make histogram the distribution we assume that (1) inside each interval all values are uniformly distributed and (2) frequencies of each value in domain should be normalized (their sum should be equal to 1).

During query evaluation histograms were read from storage and data pack is constructed of randomly chosen 2^{16} column values according to distribution described by histogram.

For the purpose of this work we assume, that each approximate query evaluation consists of 2 stages: (1) generating data according to calculated 1dim histograms and (2) evaluate the query using such prepared (approximate) table with exact engine. We created set of benchmark queries which were evaluated on data sets generated according to specific histograms. The overall rating of specific histogram depended on query results quality gained from experiments on these data sets (we will call it *approximate data*).

D. Storage Budget Discussion

As we had chosen histograms as types of data synopsis used in our framework, we should indicate how one can control existing storage budget in histogram construction. Remind that we consider budget per 1 data pack. Few question may be raised here. The most natural way to control budget in the case of histograms is to adjust number of stored buckets. This parameter affects not only storage factor but has also influence on calculation complexity (number of performed operations). The other factor may be also method of storing histogram. In general, as presented in definition, each bucket of histogram is described by left and right boundary. However with additional assumption (made also in this work) that set of histogram intervals covers whole data pack domain, apart from storing data pack minimum and maximum, to identify the bar in histogram suffice to store only its e.g. right boundary (left can be induced from prior interval). So each bucket can be stored as 2 numbers - its maximum and its frequency. Deeper optimizations of storage may depend on type of synopsis or on implementation method. In the work we focus on number of buckets as the main control parameter.

III. How to Compare Query Results?

In order to compare query results evaluated on exact (real) data and on rough data one needs to elaborate quality measures of query results. Below we describe measures used in the article. One can easily notice that they are to some extent inspired by the notions of fuzzy similarity and multi-label classification, now adopted for new purposes.

The result of every SELECT has a tabular form. We will denote it as $R = (U, C)$, where U and C are sets of its tuples and columns. U can refer to original rows or groups, possibly with limit.

In C, we distinguish columns A that are results of aggregate functions and columns G used in group by clause. Alternatively, G can gather columns that are primary key and A - all

[2]https://infobright.com/introducing-iaq/

other columns. Some columns in A can be used in ORDER BY clause. In such case the new rank function is added to C.

Let us consider two results of the same query, real result $R_r = (U_r, C)$ and an approximate result $R_a = (U_a, C)$. The idea is to check to what extent tuples in U_r and U_a match with each other with regard to columns in G and, for those tuples which can be matched, how similar are their values over A. Similarity of R_r and R_a should be within [0,1], and equal to 1 only if R_r and R_a are identical.

Consider a pair of tuples $t_r \in U_r$ and $t_a \in U_a$, such that there is $g(t_r) = g(t_a)$ for every g in G.

A score of similarity of t_r and t_a is as follows:

$$s(t_r, t_a) = \prod_{c \in A} s_c(c(t_r), c(t_a))$$

where $\prod$ denotes multiplication and:

$$s_c(c(t_r), c(t_a)) = 1 - \frac{|c(t_r) - c(t_a)|}{|c(t_r)| + |c(t_a)| + 1}$$

If rank is added, it can take a form of:

$$r(t_r, t_a) = 1 - \frac{|rank(t_r) - rank(t_a)|}{|rank(t_r)| + |rank(t_a)|}$$

While rows (and groups) from results achieved from exact query and approximate query may differ we distinguish two types of mismatched rows (groups):

- False Positives (FP) - rows (groups) which shouldn't occur but did

$$FP = card(U_a \backslash U_r) = card(U_a) - card(U_r \cap U_a)$$

- True Negatives (TN) - rows (groups) which should occur but didn't

$$TN = card(U_r \backslash U_a) = card(U_r) - card(U_r \cap U_a)$$

We define ***Aggregation Similarity*** as follows:

$$AggSim(R_r, R_a) = \frac{\sum_{t_r \in U_r, t_a \in U_a : G(t_r) = G(t_a)} s(t_r, t_a)}{card(U_r)}$$

We define ***Ranking Similarity*** as follows:

$$RankSim(R_r, R_a) = \frac{\sum_{t_r \in U_r, t_a \in U_a : G(t_r) = G(t_a)} r(t_r, t_a)}{card(U_r)}$$

And finally, we define ***Total Similarity*** as follows:

$$TotSim(R_r, R_a) = \\ = \frac{\sum_{t_r \in U_r, t_a \in U_a : G(t_r) = G(t_a)} s(t_r, t_a) \cdot r(t_r, t_a)}{card(U_r) + card(U_a \backslash U_r)}$$

In Fig. 1 there is a simple example of calculation of presented measures.

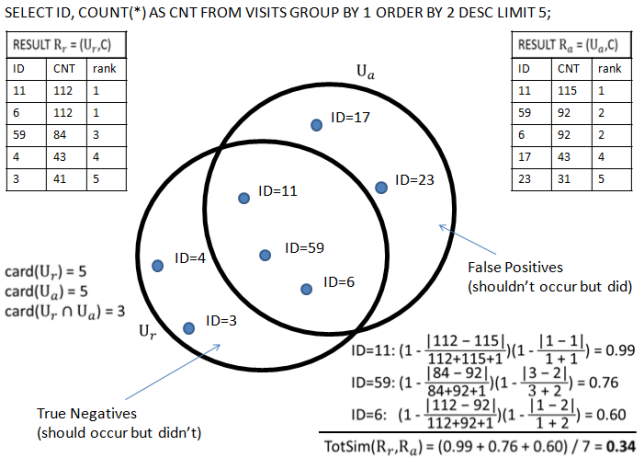

Fig. 1. Example of comparison between results of exact and approximate queries

IV. Standard Approach

A. Generation of 1-Dimensional Histogram

We had started experiments with three standard approaches to histogram generation.

As mentioned earlier we built separate histogram for each data pack (value set of 1 column from 2^{16} consecutive rows) i.e. exact data were the input to create each 1-dim histogram. After being build each histogram was stored.

Looking for the best correspondence between histogram quality measures and approximate query results quality we tested many different methods of histogram generation. Here we present results of experiments on 3 most commonly used in databases types of histograms: EquiWidth histogram (classic), EquiDepth histogram (quant) and MaxDiff histogram (diff).

Each type of histogram splits data pack range into k buckets.

- EquiDepth histogram divides the set of values into k ranges such that each range has the same number of values [10].
- EquiWidth histogram divides the set of values into k buckets of equal width [11].
- In MaxDiff histogram boundaries of intervals are chosen after analyzing differences between frequencies of adjacent bars from exact histogram (adjacency is induced by natural order of values). $k - 1$ largest differences determine split points of histograms intervals [12].

As inside each histogram's interval we assume uniform distribution for contained values, and intervals from each considered 1-dim histograms cover whole range of data pack, we may achieve in natural way from random generation values which did not occur in original data pack (False Positives). As turned out they constituted the biggest challenge.

Each type of histogram splits data pack range into k buckets, where k is established parameter corresponding to storing budget. In our experiments we took $k = 64$.

All experiments were run on the table containing 33 columns and $10 \cdot 2^{16}$ rows.

B. Standard Quality Measures for Histogram

In order to use approximate query in most efficient way not only results comparison aspect should be examined but also methods of prediction to what extent type or scale of used synopsis could affect query results quality. Such knowledge would let one to choose optimal type/size of histogram and also to control by the user threshold between storage footprint of created synopses (or on the other hand: volume of data processed during query evaluation) and query results quality. We assume that the database user is aware of the existence of such threshold.

That brings us to subproblem of defining measure of histogram quality which is correspondent to approximate query results.

We acknowledged as fundamental here the ability of generated histogram to reflect original distribution of the data pack's values. As mentioned earlier due to existing budget, in most cases, we are not able to store exact histogram of value set (such case we recognize as optimal), so such possibility is crucial to make more sophisticated inference of approximations successfully.

At the first stage we applied standard measures of similarity between distributions based on deviations from mean, which work fine for the cases without generated false positives [13], [14].

Here is the first examined measure:

$$Q1(c,p) = \sum_{v \in Dom(c)} min\{\frac{freq(p_v)}{ALL_{p_v}}, freq(v)\},$$

where c is a column, p - data pack range split (set of intervals), p_v - interval (from split) containing v, $freq(p_v)$ - frequency of p_v, $freq(v)$ - number of occurences of v and $ALL_{p_v} = end(p_v) - start(p_v) + 1$. Intuitions of measure $Q1$ are presented at Fig. 2

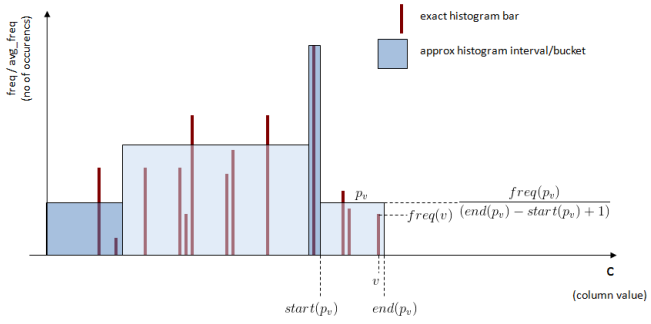

Fig. 2. Measure $Q1$ components.

The second analyzed measure was the sum of squares of deviations of exact value frequencies from corresponding bar frequency, with additional assumption that all values from intervals range is included in the sum (those which did not exists in original data have frequency equal to 0)

$$Q2(c,p) = \sum_{v \in ALL_{Dom(c)}} (freq(v) - \frac{freq(p_v)}{ALL_{p_v}})^2,$$

where c is a column, p - data pack range split (set of intervals), p_v - split interval for v, $freq(p_v)$ - frequency of interval containing v, $freq(v)$ - number of occurrences of v in data pack (non existing values have freq = 0), $ALL_{p_v} = end(p_v) - start(p_v) + 1$ and $ALL_{Dom(c)} = end(Dom(c)) - start(Dom(c)) + 1$.

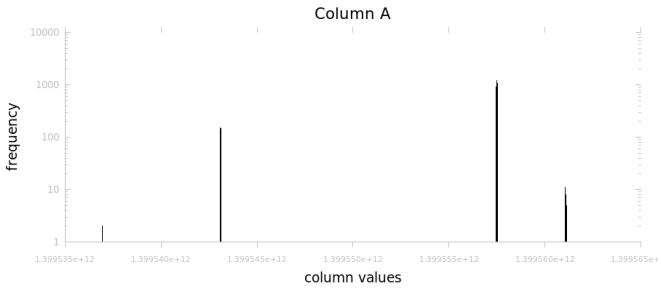

Fig. 3. Exact histogram (distribution) on 1 data pack of real-life column (Column A; logarithmic scale of frequencies)

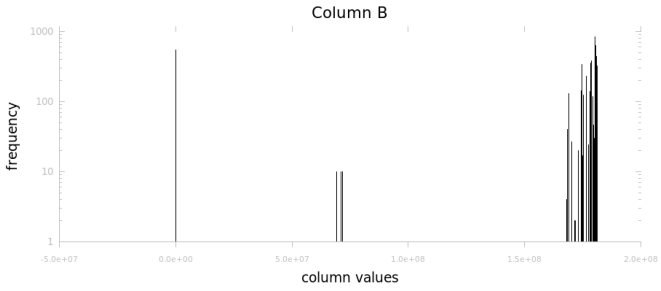

Fig. 4. Exact histogram (distribution) on 1 data pack of real-life column (Column B; logarithmic scale of frequencies)

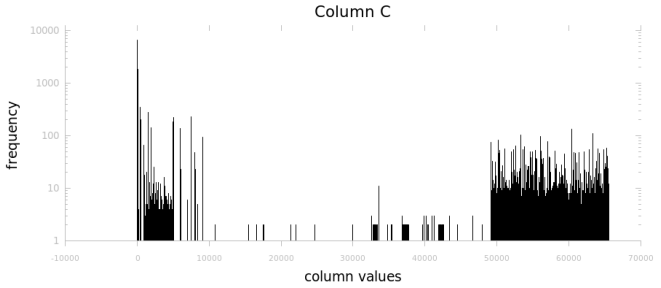

Fig. 5. Exact histogram (distribution) on 1 data pack of real-life column (Column C; logarithmic scale of frequencies)

At the first stage of experiments we calculated $Q1$ for every considered type of histograms. Next we evaluated set of prepared queries on exact data and on data generated from every histogram. We found achieved results encouraging, however we identified many hard cases for presented approach.

Lack of satisfactory correspondence between $Q1$ and query results quality for each histogram was visible both on simplest single-column queries and more complex queries. Fig. 6–9 illustrate results of testing query that simulate calculation of exact distribution of column values:

```
SELECT col, count(*) FROM t GROUP BY col
```

We chose exemplary columns from real data set and mark them as A, B, C. Their real distributions are presented on Fig. 3–5.

Calculated value of $Q1$ was additionally divided by number of rows in table t. The higher the value of $Q1$ the better quality of the histogram.

Column	Metric	Diff	Classic	Quant
A	Q1 Measure	0.8620	0.0601	0.8460
	TotSim	0.0281	0.0014	0.0351
B	Q1 Measure	0.1086	0.0004	0.0925
	TotSim	0.0036	0.0002	0.0075
C	Q1 Measure	0.4812	0.3612	0.3674
	TotSim	0.4562	0.5192	0.4520

Fig. 6. Illustration of lack of satisfactory correspondence between $Q1$ and quality of approximate queries results. Diff, Classic and Quant stand for MaxDiff, EquiWidth and EquiDepth histograms respectively. Values of $Q1$ are averages from 10 data packs.

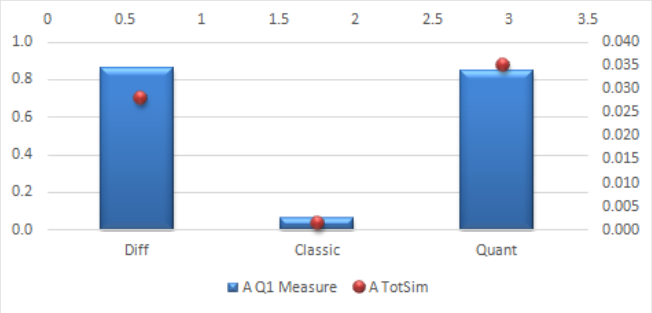

Fig. 7. Lack of satisfactory correspondence between $Q1$ and quality of approximate query on column A

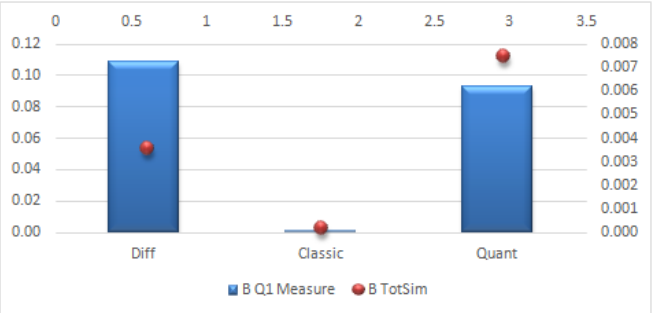

Fig. 8. Lack of satisfactory correspondence between $Q1$ and quality of approximate query on column B

Similar tests were run for $Q2$. Here also observed correspondence between histogram quality and quality of approximate query results was disappointing. Results are presented

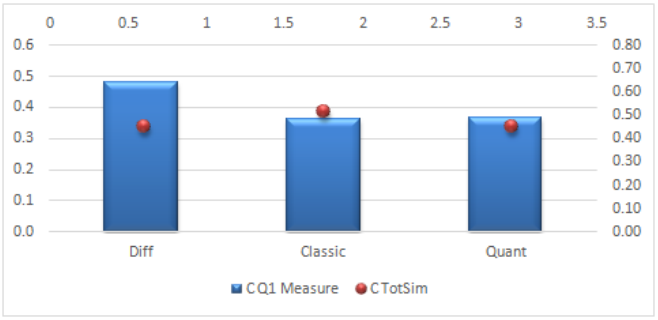

Fig. 9. Lack of satisfactory correspondence between $Q1$ and quality of approximate query on column C

on figures 10–13. Analogously value of $Q2$ was divided by number of rows in the table. In contrast to the measure $Q1$, the higher the value of $Q2$, the worse the quality of the histogram.

Column	Metric	Diff	Classic	Quant
A	Q2 Measure	330.669	981.971	351.434
	TotSim	0.028	0.001	0.035
B	Q2 Measure	2 785.017	2 837.871	2 821.522
	TotSim	0.004	0.0002	0.007
C	Q2 Measure	4 446.830	5 694.313	5 587.474
	TotSim	0.456	0.519	0.452

Fig. 10. Table illustrating lack of satisfactory correspondence between $Q2$ and quality of approximate queries (on data generated according to specified histogram)

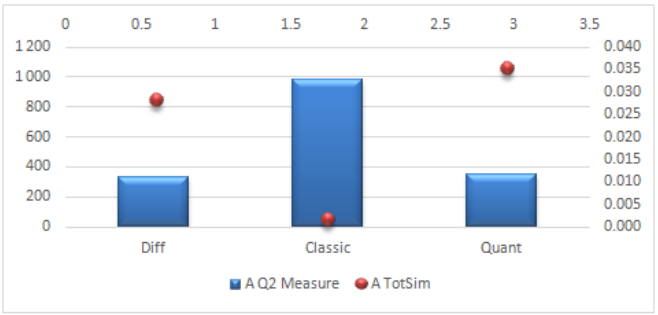

Fig. 11. Lack of satisfactory correspondence between $Q2$ and quality of approximate query on column A

V. Specific Histogram Quality Measures

Because of discrepancies between both $Q1$, $Q2$ and approximate query results quality, some new histogram quality measures had to be developed. Discrepancies analysis revealed that the main reason of inadequacy of applied measures was assumption of uniform distribution of values frequencies inside each interval. In particular no information of gaps in data packs domain was involved. By *gap* (if exists) we take interval between two consecutive values that exists in original data set. More formally:

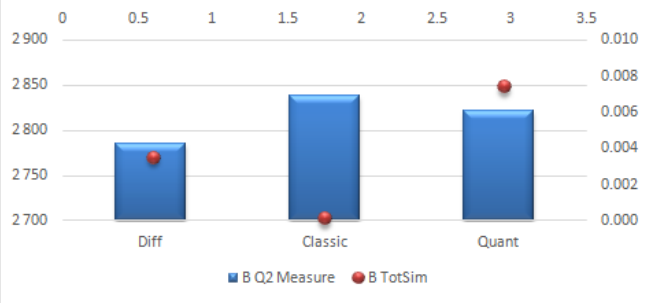

Fig. 12. Lack of satisfactory correspondence between $Q2$ and quality of approximate query on column B

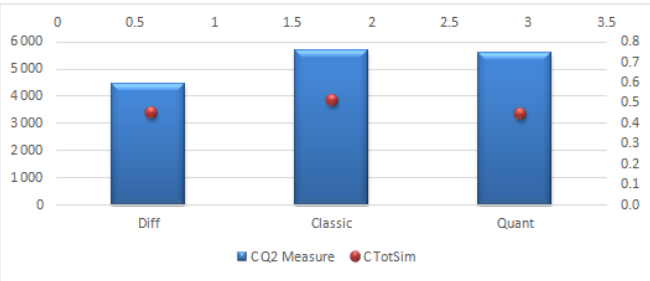

Fig. 13. Lack of satisfactory correspondence between $Q2$ and quality of approximate query on column C

Definition. Let $v, w \in Dom(c)$ be two consecutive (in the sense of linear order on values of column c) values that exist in considered value set. Then by *gap* we take set $\{x \in ALL_{Dom(c)} : x > v \wedge x < w\}$

As pointed out earlier because of method of data generation from considered types of histograms, one is exposed to generating false-positive cases. Gaps (especially wide ones), present in original data set would cause deterioration of quality of approximate query results in two ways. First – due to numerous false-positive cases and in the consequence also true-negative cases – it will reduce Total Similarity value. On the other hand too voluminous domain of interval makes average frequency lower and as a result may decrease Aggregation Similarity.

After testing some preliminar candidates, we introduced measure of the histogram which indicates difficulty of gaining proper correspondence between histogram quality and approximate query quality:

$$Q_{HG}(c,p) = \sum_{i=1}^{p_{cnt}} freq(p_i) \cdot FALSE(p_i)$$

, where c stands for column, p - split (set of intervals), p_{cnt} - number of intervals, $freq(p_i)$ - frequency of ith interval, and $FALSE(p_i)$ - number of false-positives generated in ith interval (which does not exist in original data set).

Q_{HG} was divided by number of rows in table t. The higher the value of Q_{HG}, the worse the quality of the histogram. Figures 14–16 show that Q_{HG} does not correspond to approximate query quality in direct way, however we may notice some regularities. First, calculated value of Q_{HG} explains poor quality of query results for columns A and B. For column C considered query returned better approximations and corresponding value of Q_{HG} was significantly lower than for A and B.

Column	Metric	Diff	Classic	Quant
A	Q_{HG} Measure	66 441 005.00	13 120 751.37	71 920 778.98
	TotSim	0.028	0.001	0.035
B	Q_{HG} Measure	71 474 878.37	36 296 002.79	60 186 447.39
	TotSim	0.004	0.0002	0.007
C	Q_{HG} Measure	1 087.03	588.48	859.77
	TotSim	0.456	0.519	0.452

Fig. 14. Illustration of the rule: high value of Q_{HG} corresponds to poor approximate query results quality

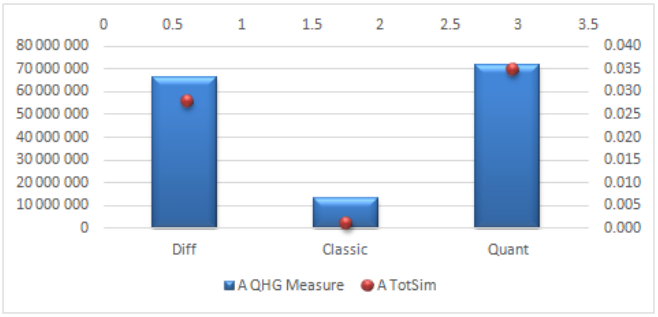

Fig. 15. Illustration of the rule: high value of Q_{HG} corresponds to poor approximate query results quality - for column A

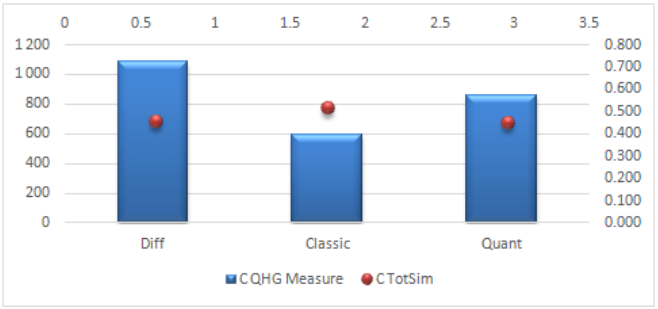

Fig. 16. Illustration of the rule: low value of Q_{HG} corresponds to well-approximated query results quality - for column C

To confirm hypothesis of correspondence between high value Q_{HG} and poor query results, we perform more experiments. We defined importance ranking formula for gaps and added to synopsis information of 100 most important gaps per data pack.

Definition. Gap importance ranking for each data pack (of column c) with given split p is defined as follow:

$$RANK\_GAP_{c,p}(gap) = freq(i(gap)) \cdot FALSE(gap),$$

where $freq$ stands for interval frequency, $i(gap)$ stands for index of interval containing gap , and $FALSE(gap)$ stands for number of false-positives covered by gap.

We consider also the case where data packs differ between themselves w.r.t. contained gaps. We observed such situation in real life data. In such case we may consider aggregated budget for gaps for several e.g. consecutive data packs (not per single data pack like in assumptions made at the beginning). Such modification enable to assign parts of the budget to data packs in more flexible way (bigger part to data packs with a lot of gaps, smaller - to the data packs with a few). In next series of experiments instead of 100 most important gaps per data pack we add information of 1000 most important gaps but for 10 data packs (whole column value set). Importance we calculate using formula:

Definition. Holistic gap importance ranking for column c and given split p is defined as follow:

$$BIGRANK\_GAP_{c,p}(gap) = \\ = \sum_{j=1}^{10} freq_j(i(gap)_j) \cdot FALSE_j(gap),$$

where $freq_j$ stands for frequency of interval in jth data pack, $i(gap)_j$ stands for index of interval in jth data pack containing gap, and $FALSE_j(gap)$ stands for number of false-positives covered by gap in jth data pack.

At Fig. 17 we presented dependency between order of magnitude of Q_{HG} and approximate query results quality. For clarity sake dependency was shown only for MaxDiff histogram (for other types dependency is the same). We can observe that if value of Q_{HG} is not low there is no chance to achieve good results of approximate query. However relatively small values of Q_{HG} would not guarantee good quality of approximate query results, and strength of correspondence is depended on chosen column. Therefore, we conclude there are some other factors that affect approximate query results quality.

Column	Metric	Diff	RANK Diff	BIGRANK Diff
A	Q_{HG} Measure	66 441 005.00	14.02	19.95
	TotSim	0.028	0.489	0.713
B	Q_{HG} Measure	71 474 878.37	206 719.29	844.31
	TotSim	0.004	0.011	0.171
C	Q_{HG} Measure	1 087.03	553.17	561.79
	TotSim	0.49	0.59	0.69

Fig. 17. Correspondence between order of magnitude of value Q_{HG} and approximate query results quality.

VI. Conclusions and Future Work

In the article we present preliminary method for calculation of quality of histogram representing original column values in data set. Such measure should in the assumption correspond to quality of approximate query run over data generated from the histogram. Development of such measure would allow at the loading stage to construct histogram reflecting the input data in the best way with limited storage budget maintained.

Performed experiments confirm that many features of input histograms may have influence on quality of approximate query results. Some of them are not identified yet. We suspect such characteristics might be related to intervals width or - like in case of gaps - to distributions of exact values frequencies inside each interval. Probably most of these factors may be expressible in terms of statistical measures of dispersion, symmetry or skewness. We can adapt them to histogram quality formulas however proper choice and the way of applying chosen measure requires much more experimental work.

References

[1] V. Ganti, M.-L. Lee, and R. Ramakrishnan, "Icicles: Self-tuning samples for approximate query answering." in *VLDB*, vol. 176. Citeseer, 2000, p. 187.

[2] S. Chaudhuri, G. Das, and V. Narasayya, "Optimized Stratified Sampling for Approximate Query Processing," *ACM Trans. Database Syst.*, vol. 32, no. 2, p. 9, 2007.

[3] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets," *VLDB J.*, vol. 10, no. 2-3, pp. 199–223, 2001.

[4] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends in Databases*, vol. 4, no. 1-3, pp. 1–294, 2012. doi: 10.1561/1900000004. [Online]. Available: http://dx.doi.org/10.1561/1900000004

[5] B. Mozafari and N. Niu, "A handbook for building an approximate query engine," *IEEE Data Eng. Bull.*, vol. 38, no. 3, pp. 3–29, 2015. [Online]. Available: http://sites.computer.org/debull/A15sept/p3.pdf

[6] D. Ślęzak and V. Eastwood, "Data warehouse technology by infobright," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 841–846.

[7] D. Ślęzak, P. Synak, A. Wojna, and J. Wróblewski, "Two database related interpretations of rough approximations: Data organization and query execution," *Fundam. Inform.*, vol. 127, no. 1-4, pp. 445–459, 2013. doi: 10.3233/FI-2013-920. [Online]. Available: http://dx.doi.org/10.3233/FI-2013-920

[8] D. Ślęzak and M. Kowalski, "Towards approximate SQL–Infobright's approach," in *Rough Sets and Current Trends in Computing*. Springer, 2010, pp. 630–639.

[9] M. Kowalski, D. Ślęzak, and P. Synak, "Approximate assistance for correlated subqueries," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Kraków, Poland, September 8-11, 2013.*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2013, pp. 1455–1462. [Online]. Available: http://fedcsis.org/2013/

[10] S. Chaudhuri, "An overview of query optimization in relational systems," in *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 1998, pp. 34–43.

[11] R. P. Kooi, "The optimization of queries in relational databases," 1980.

[12] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption," in *VLDB*, vol. 97, 1997, pp. 486–495.

[13] Y. E. Ioannidis and V. Poosala, "Balancing histogram optimality and practicality for query result size estimation," in *ACM SIGMOD Record*, vol. 24, no. 2. ACM, 1995, pp. 233–244.

[14] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal histograms with quality guarantees," in *VLDB*, vol. 98, 1998, pp. 24–27.