Decision Support Systems 52 (2011) 82–94

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Efficient construction of histograms for multidimensional data using quad-trees

Yohan J. Roh [a,*], Jae Ho Kim [b], Jin Hyun Son [c], Myoung Ho Kim [b]

[a] Data Analytics Group, Samsung Advanced Institute of Technology, Samsung Electronics Nongseo-dong, Yongin Si Giheung-gu, Gyeonggi-Do 446-712, South Korea
[b] Department of Computer Science KAIST 373-1 Guseong-dong, Yuseong-gu, Taejon 305–701, South Korea
[c] Department of Computer Science and Engineering Hanyang University 1271 Sa-1 dong, Ansan, Kyunggi-do 425-791, South Korea

ARTICLE INFO

Article history:
Received 29 January 2010
Received in revised form 2 May 2011
Accepted 15 May 2011
Available online 19 May 2011

Keywords:
Data management
Query optimization
Selectivity estimation
Multidimensional histograms

ABSTRACT

Histograms can be useful in estimating the selectivity of queries in areas such as database query optimization and data exploration. In this paper, we propose a new histogram method for multidimensional data, called the Q-Histogram, based on the use of the quad-tree, which is a popular index structure for multidimensional data sets. The use of the compact representation of the target data obtainable from the quad-tree allows a fast construction of a histogram with the minimum number of scanning, i.e., only one scanning, of the underlying data. In addition to the advantage of computation time, the proposed method also provides a better performance than other existing methods with respect to the quality of selectivity estimation. We present a new measure of data skew for a histogram bucket, called the weighted bucket skew. Then, we provide an effective technique for skew-tolerant organization of histograms. Finally, we compare the accuracy and efficiency of the proposed method with other existing methods using both real-life data sets and synthetic data sets. The results of experiments show that the proposed method generally provides a better performance than other existing methods in terms of accuracy as well as computational efficiency.

1. Introduction

With increasing data volumes, the performance demands on database systems have grown and the need to produce accurate approximations of data distributions has also increased significantly. In particular, the estimation of the selectivity of a query, i.e., the number of data objects in the query region, can be used for database query optimization [18,19]. It also can be used for some types of query processing such as skyline query processing, spatio-temporal query processing, top-k query processing and so on [2,4–7,27,32–34].

Motivated by these applications, there has been much work on the problem of selectivity estimation: histograms [1,3,9–11,13–16,20,25,29,30,35], wavelet transformation [24,36], discrete cosine transformation [23], and sampling [17]. Among these approaches, histograms have been shown to be one of the most popular and effective ways to obtain accurate estimates of selectivity for multi-dimensional queries [10].

A histogram consists of a set of buckets b_i, $i=1, \ldots, n$, where each bucket b_i has its data space s_i and the number of data objects f_i in s_i. All the data objects in the region of a bucket are assumed to be uniformly distributed (commonly called *uniform distribution assumption*). The number of buckets is usually a system parameter and is reasonably small so that all the buckets can be kept in memory. The process of constructing a histogram is typically performed periodically to reflect changes in the underlying data distribution.

Given a data range I specified by a query, an estimate of the selectivity for the query is computed as follows, under *uniform distribution assumption*: $\sum_{i=1 \ldots n} |s_i \wedge I| / |s_i| \cdot f_i$. Here, $|\,|$ denotes the size of a data space and '$s_i \wedge I$' denotes the intersection of s_i and I. An estimate of the selectivity for one bucket is computed in proportion to the size of the overlapping region between the query region and the bucket region. The selectivity estimate for a query is the sum of all the estimated values for all the buckets.

When data objects are not uniformly distributed in buckets, the accuracy of histograms will decrease. Therefore, a histogram should be organized in such a way that data in each bucket is as uniformly distributed as possible.

Now let us consider index structures that are used widely in commercial database systems. As noted in [10,19], some of the existing index structures can be an interesting starting point for constructing histograms. The quad-tree and its variants have been popularly used as index structures for the fast access of multidimensional data sets. When a quad-tree has already been used as an index for some applications, we can improve the cost of histogram construction by using the data partition information implied in this quad-tree. That is, utilizing the existing quad-tree can provide an advantage of computing time for construction of buckets. Then, our problem can be stated simply as follows, when a given number of buckets is B: Partition a set of leaf nodes in a quad-tree into B groups such that the data objects in each group are as uniformly distributed as possible. Here, each group corresponds to one bucket. When there are K leaf

* Corresponding author. Tel.: +82 312809728; fax: +82 312809086.
E-mail addresses: yohan.roh@samsung.com (Y.J. Roh), jaeho@dbserver.kaist.ac.kr (J.H. Kim), jhson@hanyang.ac.kr (J.H. Son), mhkim@dbserver.kaist.ac.kr (M.H. Kim).

0167-9236/$ – see front matter. Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.05.006

nodes in the quad-tree, the number of ways of partitioning K leaf nodes into B groups, commonly known as *Stirling number of the second kind* $S(K, B)$, can be quite large in practice. This problem is *NP-Hard*, and therefore, some heuristics need to be employed. The partitioning of the leaf nodes can proceed in either a bottom-up or top-down fashion. We will start with the root node of the quad-tree and proceed in a top-down fashion.

In this paper, we will propose a new multidimensional histogram method, called the Q-Histogram, which is based on the use of the existing quad-tree. The proposed Q-Histogram divides a given data set with various levels of granularity by using the information in the quad-tree with the minimum number of scanning, i.e., only one scanning, of the underlying data. Through extensive experiments, we show that Q-Histogram has better performance than other existing methods with respect to accuracy as well as computational efficiency.

The rest of the paper is organized as follows. Section 2 describes related work. We present our proposed histogram method in Section 3. Section 4 provides the results of performance experiments with four real-life data sets as well as one synthetic data set. Finally in Section 5, we present conclusions and future work.

2. Related work

Histograms on multiple attributes can be used for processing and optimizing queries. For query optimization, histograms can be used to estimate the selectivity of queries and to generate the most efficient query execution plans [18,19].

For skyline query processing, Chaudhuri et al. [6] and Papadias et al. [27] use histograms to accurately estimate the result sizes of skyline queries, which can be useful in providing immediate feedback to the user and implementing skyline computation as an operator in database systems.

For spatio-temporal query processing, the authors of [7] use a histogram technique and extend it with velocities to estimate the selectivity of spatio-temporal window queries, i.e., the number of objects that will appear in the query window at a given future time. For the same purpose, Tao et al. [33] propose a set of histogram-based solutions. Sun et al. [32] make use of histograms to accurately estimate the selectivity of spatio-temporal joins, i.e., for two given sets S_1 and S_2 of objects, the number of pairs $<o_1, o_2>$ of objects, such that $o_1 \in S_1$, $o_2 \in S_2$, and the distance between these two objects at a given future time is below a certain threshold.

For load-balancing of parallel hash joins, Poosala and Ioannidis [28] use the statistics of histograms to accurately estimate the cost required to perform the join operation, and effectively balance the load across nodes that participate in the parallel execution.

For top-k query processing, Bruno et al. [4] and Chaudhuri et al. [5] use histograms for translating a top-k request into a single range query that can be efficiently processed by existing database engines. They have shown that using histograms can avoid the requirement of a full sequential scan of the database, and thus significantly reduce the time to perform top-k queries.

Over the past decades, many studies have been conducted to enhance the performance of multidimensional histograms. The underlying assumption in using a histogram is that the histogram performs well when data is uniformly distributed in every bucket. However, the problem of organizing buckets in such a way that the data is uniformly distributed in every bucket is *NP-hard* in two or more dimensions [26] and heuristics have been proposed.

The *EquiDepth* histogram method [25] partitions the target space, one dimension at a time. Here, in each i-th dimension, the target space is divided into v_i intervals, each of which has the same number of data. So, for a d-dimensional data set, a set of $v_1 \times v_2 \times \ldots \times v_d$ buckets is constructed, where each bucket contains the same number of data. The *EquiDepth* histogram may be faster to construct among other types of histograms, while because of its rigid structure it may not be flexible to cope with various cases of data skew.

The *MinSkew* histogram method [1] uses binary space partitioning, where a bucket is partitioned into two new buckets. This partitioning approach may construct histograms rapidly; however, *MinSkew* may not recognize regions where data are not uniformly distributed, which may decrease the accuracy of selectivity estimation. This is because the partitioning heuristics of *MinSkew* is based on data skew in only one-dimension at a time rather than considering the skew of multiple dimensions at once.

The *GenHist* histogram method [15,16] uses multidimensional grids of various sizes, where high-frequency grid cells are converted into buckets. More specifically, *GenHist* iteratively constructs a certain number of buckets by using grids. Here, the grid sizes and the number of buckets constructed per iteration are determined by using the system parameter (i.e., the total number of buckets) and the user-provided parameter (i.e., initial grid size). Being different from the above approaches, this method directly approximates multidimensional (i.e., joint) data distributions. The authors of *GenHist* claim that the *GenHist* histogram behaves more accurately for data sets in *high-dimensional* spaces than some previous approaches, such as random sampling, wavelet transformation [36], and *MinSkew* [1]. However, the performance of *GenHist* may vary depending on the input parameters. Furthermore, in practice, it is difficult for users to provide the optimal or a near optimal value for the required parameter. Another drawback of this technique is that it requires multiple passes (at least 5 to 10) over the entire data set [3].

The *RK-Hist* histogram method, which has been recently proposed in [10], uses a variant of an R-tree index, called the *Hilbert packed R-tree* [21], where the entire data are sorted based on their own positions along the Hilbert curve. The sorted data are divided into several leaf nodes of the tree, in which the size of each leaf node is a disk block. Then, *RK-Hist* creates an initial set of buckets, each constructed by merging a fixed number of leaf nodes. For each bucket, the skew of data is computed, and then some bucket with a high skew is split into two new buckets repeatedly, until the total number of buckets becomes the predefined number or there is no improvement of the total skew of data in buckets. The authors of *RK-Hist* claim that the *RK-Hist* histogram works better than other existing methods, such as a traditional histogram technique [29], *EquiDepth* [25], and *GenHist* [15,16], in terms of estimation accuracy. However, the worst case time complexity of *RK-Hist* is $O(d \cdot N^2)$, where d is the dimension of the data space and N is the number of data objects. That is, the construction time of *RK-Hist* will be high, when the number of data becomes large. *RK-Hist* may introduce unnecessary buckets, when a fixed number (say p) of leaf nodes are merged into an initial bucket. For example, consider a nonleaf node u with a very low skew that is an ancestor of a large number of leaf nodes. If the number of the descendant leaf nodes of u is much greater than p, several buckets will be constructed from these leaf nodes, but only one bucket consisting of a single node u suffices to provide accurate selectivity estimation instead of several buckets. Note that, after the initial buckets are made, no merging is performed in subsequent steps.

There are several approaches for the layout of buckets. In the grid approach, buckets are arranged in rows and columns (e.g., as in the well-known equal-width histogram). In the recursively partitioning approach, a bucket is recursively partitioned into two new buckets along some dimension (e.g., as in *MinSkew* [1]). There are also other approaches that impose fewer restrictions than the above approaches on the arrangement of buckets, that is, allow a newly created bucket to cover a portion of data space in a more flexible way. For example, in *GenHist* [15,16] and *RK-Hist* [10], the regions of buckets are allowed to overlap. The histogram method proposed in this paper also allows the regions of buckets to overlap.

Histograms are typically recomputed to reflect updates of the underlying data in a periodic manner. There is another interesting approach in maintaining histograms, called the *self-tuning histogram*