Histograms for OLAP and Data-Stream Queries

Francesco Buccafurri
DIMET, Università di Reggio Calabria, Italy

Gianluca Caminiti
DIMET, Università di Reggio Calabria, Italy

Gianluca Lax
DIMET, Università di Reggio Calabria, Italy

INTRODUCTION

Histograms are an important tool for data reduction both in the field of data-stream querying and in OLAP, since they allow us to represent large amount of data in a very compact structure, on which both efficient mining techniques and OLAP queries can be executed. Significant time- and memory-cost advantages may derive from data reduction, but the trade-off with the accuracy has to be managed in order to obtain considerable improvements of the overall capabilities of mining and OLAP tools.

In this chapter we focus on histograms, that are shown in the recent literature to be one of the possible concrete answers to the above requirements.

BACKGROUND

Data synopses are widely exploited in many applications. Every time it is necessary to produce fast query answers and a certain estimation error can be accepted, it is possible to inquire summary data rather than the original ones and to perform suitable interpolations. This happens for example in OLAP, where a typical query is a *range query*, or in the case of continuous query over data streams.

A possible solution to this problem is using sampling methods (Gemulla, Lehner, & Haas, 2007; Gryz, Guo, Liu & Zuzarte, 2004): only a small number of suitably selected records of R, well *representing* R, are stored. The query is then evaluated by exploiting these samples instead of the full relation R. Sampling techniques are very easy to implement.

Regression techniques try to model data as a function in such a way that only a small set of coefficients representing such a function is stored, rather than the original data. The simplest regression technique is the linear one, modeling a data distribution as a linear function. Despite its simplicity, not allowing to capture complex relationships among data, this technique often produces acceptable results. There are also non-linear regressions, significantly more complex than the linear one from the computational point of view, yet applicable to a much larger set of cases.

Besides these techniques, another possible solution relies on the usage of histograms.

MAIN THRUST OF THE CHAPTER

Histograms are a lossy compression technique widely used in various application contexts, like query optimization, statistical and temporal databases and OLAP applications. In OLAP, compression allows us to obtain fast approximate answers by evaluating queries on reduced data in place of the original ones. Histograms are well-suited to this purpose, especially in case of range queries (Muthukrishnan & Strauss, 2003).

A histogram is a compact representation of a relation R. It is obtained by partitioning an attribute X of the relation R into k sub-ranges, called buckets, and by maintaining for each of them a few information, typically corresponding to the bucket boundaries, the number of tuples with value of X belonging to the sub-range associated to the bucket (often called *sum of the bucket*), and the number of distinct values of X of such a sub-range occurring in some tuple of R (i.e., the number of non-null frequencies of the sub-range).

Figure 1 reports an example of 3-bucket histogram, built on a domain of size 12 with 3 null elements. For each bucket (represented as an oval), we have reported the boundaries (on the left and the right side, respectively) and the value of the sum of the elements belonging to the bucket (inside the oval). Observe that, the null values (i.e. the values at 6, 7 and 9) do not occur in any bucket.

A range query, defined on an interval I of X, evaluates the number of occurrences in R with value of X in I. Thus, buckets embed a set of pre-computed disjoint range queries capable of covering the whole active domain of X in R (by "active" here we mean attribute values actually appearing in R). As a consequence, the histogram does not give, in general, the possibility of evaluating exactly a range query not corresponding to one of the pre-computed embedded queries. In other words, while the contribution to the answer coming from the sub-ranges coinciding with entire buckets can be returned exactly, the contribution coming from the sub-ranges which partially overlap buckets can be only estimated, since the actual data distribution inside the buckets is not available. For example, concerning the histogram shown in Figure 1, a range query from 4 to 8 is estimated by summing (1) the partial contribution of bucket 1 computed by CVA (see Section "Estimation inside a bucket"), that is 104.8, and (2) the *sum* of bucket 2, that is 122. As a consequence, the range query estimation is 226.8 whereas the exact result is 224.

Constructing the best histogram means defining the boundaries of buckets in such a way that the estimation of the non pre-computed range queries becomes more effective (e.g., by avoiding that large frequency differences arise inside a bucket). This approach corresponds to finding, among all possible sets of pre-computed range queries, the set which guarantees the best estimation of the other (non pre-computed) queries, once a technique for estimating such queries is defined.

Besides this problem, which we call the *partition problem*, there is another relevant issue to investigate: How to improve the estimation inside the buckets? We discuss about both the above issues in the following two sections.

The Partition Problem

This issue has been widely analyzed in the past and a number of techniques have been proposed. Among these, we first consider the Max-Diff histogram and the V-Optimal histogram. Even though they are not the most recent techniques, we deeply cite them since they are still considered points of reference.

We start by describing the Max-Diff histogram.

Let $V=\{v_1, \ldots, v_n\}$ be the set of values of the attribute X actually appearing in the relation R and $f(v_i)$ be the number of tuples of R having value v_i in X. A Max-Diff histogram with h buckets is obtained by putting a boundary between two adjacent attribute values v_i and v_{i+1} of V if the difference between $f(v_{i+1}) \cdot s_{i+1}$ and $f(v_i) \cdot s_i$ is one of the $h-1$ largest such differences (where s_i denotes the spread of v_i, that is the distance from v_i to the next non-null value).

A V-Optimal histogram, which is the other classical histogram we describe, produces more precise results than the Max-Diff histogram. It is obtained by selecting the boundaries for each bucket i so that the query approximation error is minimal. In particular, the boundaries of each bucket i, say lb_i and ub_i (with $1 \leq i \leq h$, where h is the total number of buckets), are fixed in such a way that:

$$\sum_{i=1}^{h} SSE_i$$

is minimum, where the standard squared error of the i-th bucket

Figure 1. An example of a 3-bucket histogram built on a domain of size 12.