Approximate Range Queries by Histograms in OLAP

Francesco Buccafurri
University "Mediterranea" of Reggio Calabria, Italy

Gianluca Lax
University "Mediterranea" of Reggio Calabria, Italy

INTRODUCTION

Online analytical processing applications typically analyze a large amount of data by means of repetitive queries involving aggregate measures on such data. In fast OLAP applications, it is often advantageous to provide approximate answers to queries in order to achieve very high performances. A way to obtain this goal is by submitting queries on compressed data in place of the original ones. Histograms, initially introduced in the field of query optimization, represent one of the most important techniques used in the context of OLAP for producing approximate query answers.

BACKGROUND

Computing aggregate information is a widely exploited task in many OLAP 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. The typical OLAP query is the range query.

The range query estimation problem in the one-dimensional case can be stated as follows: given an attribute X of a relation R, and a range I belonging to the domain of X, estimate the number of records of R with value of X lying in I. The challenge is finding methods for achieving a small estimation error by consuming a fixed amount of storage space.

A possible solution to this problem is using sampling methods; only a small number of suitably selected records of R, representing R well, are stored. The range query is then evaluated by exploiting this sample instead of the full relation R. Recently, Wu, Agrawal, and Abbadi (2002) have shown that in terms of accuracy, sampling techniques based on the cumulative distribution function are definitely better than the methods based on tuple sampling (Chaudhuri, Das & Narasayya, 2001; Ganti, Lee & Ramakrishnan, 2000). The main advantage of sampling techniques is that they are very easy to implement.

Besides sampling, 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, which models a data distribution as a linear function. Despite its simplicity, not allowing the capture of 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, but applicable to a much larger set of cases.

Another possibility for facing the range query estimation problem consists of using wavelets-based techniques (Chakrabarti, Garofalakis, Rastogi & Shim, 2001; Garofalakis & Gibbons, 2002; Garofalakis & Kumar, 2004). Wavelets are mathematical transformations storing data in a compact and hierarchical fashion used in many application contexts, like image and signal processing (Kacha, Grenez, De Doncker & Benmahammed, 2003; Khalifa, 2003). There are several types of transformations, each belonging to a family of wavelets. The result of each transformation is a set of values, called *wavelet coefficients*. The advantage of this technique is that, typically, the value of a (possibly large) number of wavelet coefficients results to be below a fixed threshold, so that such coefficients can be approximated by 0. Clearly, the overall approximation of the technique as well as the compression ratio depends on the value of such a threshold. In the last years, wavelets have been exploited in data mining and knowledge discovery in databases, thanks to time and space efficiency and data hierarchical decomposition characterizing them. For a deeper treatment about wavelets, see Li, Li, Zhu, and Ogihara (2002).

Besides sampling and wavelets, histograms are used widely for estimating range queries. Although sometimes wavelets are viewed as a particular class of histograms, we prefer to describe histograms separately.

MAIN THRUST

Histograms are a lossy compression technique widely applied 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 the case of range queries.

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

Recall that 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 precomputed disjoint range queries capable of covering the whole active domain of X in R (here, active means attribute values actually appearing in R). As a consequence, the histogram, in general, does not give the possibility of evaluating exactly a range query not corresponding to one of the precomputed embedded queries. In other words, while the contribution to the answer coming from the subranges coinciding with entire buckets can be returned exactly, the contribution coming from the subranges that partially overlap buckets can only be estimated, since the actual data distribution is not available.

Constructing the best histogram thus may mean defining the boundaries of buckets in such a way that the estimation of the non-precomputed 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 precomputed range queries, the set that guarantees the best estimation of the other (non-precomputed) 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 both of these issues in the following two sections.

The Partition Problem

This issue has been analyzed widely in the past, and a number of techniques has 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 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 MaxDiff 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 $\Sigma_i SSE_i$ is minimal, where SSE_i is the standard squared error of the bucket i-th.

V-Optimal histogram uses a dynamic programming technique in order to find the optimal partitioning w.r.t. a given error metrics. Even though the V-Optimal histogram results more accurate than Max-Diff, its high space and time complexities make it rarely used in practice.

In order to overcome such a drawback, an approximate version of the V-Optimal histogram has been proposed. The basic idea is quite simple. First, data are partitioned into l disjoint chunks, and then the V-Optimal algorithm is used in order to compute a histogram within each chunk. The consequent problem is how to allocate buckets to the chunks such that exactly B buckets are used. This is solved by implementing a dynamic programming scheme. It is shown that an approximate V-Optimal histogram with $B + l$ buckets has the same accuracy as the non-approximate V-Optimal with B buckets. Moreover, the time required for executing the approximate algorithm is reduced by multiplicative factor equal to $1/l$.

We call the histograms so far described *classical histograms*.

Besides accuracy, new histograms tend to satisfy other properties in order to allow their application to new environments (e.g., knowledge discovery). In particular, (1) the histogram should maintain in a certain measure the semantic nature of original data, in such a way that meaningful queries for mining activities can be submitted to reduced data in place of original ones. Then, (2) for a given kind of query, the accuracy of the reduced structure should be guaranteed. In addition, (3) the histogram should efficiently support hierarchical range queries in order not to limit too much the capability of drilling down and rolling up over data.