Data Cube Compression Techniques: A Theoretical Review

Alfredo Cuzzocrea
University of Calabria, Italy

INTRODUCTION

OnLine Analytical Processing (OLAP) research issues (Gray, Chaudhuri, Bosworth, Layman, Reichart & Venkatrao, 1997) such as data cube modeling, representation, indexing and management have traditionally attracted a lot of attention from the Data Warehousing research community. In this respect, as a fundamental issue, the problem of *efficiently compressing the data cube* plays a leading role, and it has involved a vibrant research activity in the academic as well as industrial world during the last fifteen years.

Basically, this problem consists in dealing with massive-in-size data cubes that make access and query evaluation costs prohibitive. The widely accepted solution consists in generating a *compressed representation* of the target data cube, with the goal of reducing its size (given an input space bound) while admitting loss of information and approximation that are considered irrelevant for OLAP analysis goals (e.g., see (Cuzzocrea, 2005a)). Compressed representations are also referred-in-literature under the term "*synopsis data structures*", i.e. succinct representations of original data cubes introducing a limited loss of information. Benefits deriving from the data cube compression approach can be summarized in a relevant reduction of computational overheads needed to both represent the data cube and evaluate resource-intensive OLAP queries, which constitute a wide query class allowing us to extract useful knowledge from huge amounts of multidimensional data repositories in the vest of summarized information (e.g., *aggregate information* based on popular SQL aggregate operators such as SUM, COUNT, AVG etc), otherwise infeasible by means of traditional OLTP approaches. Among such queries, we recall: (*i*) *range-queries*, which extract a sub-cube bounded by a given range; (*ii*) *top-k queries*, which extract the *k* data cells (such that *k* is an input parameter) having the highest aggregate values; (*iii*) *iceberg queries*, which extract the data cells having aggregate values above a given threshold.

This evidence has given raise to the proliferation of a number of *Approximate Query Answering* (AQA) techniques, which, based on data cube compression, aim at providing *approximate answers* to resource-intensive OLAP queries instead of computing exact answers, as decimal precision is usually negligible in OLAP query and report activities (e.g., see (Cuzzocrea, 2005a)).

Starting from these considerations, in this article we first provide a comprehensive, rigorous survey of main data cube compression techniques, i.e. histograms, wavelets, and sampling. Then, we complete our analytical contribution with a detailed theoretical review focused on spatio-temporal complexities of these techniques.

BACKGROUND

A *data cube* $\mathcal{L}$ is a tuple $\mathcal{L} = \langle \mathcal{C},\mathcal{J},\mathcal{H},\mathcal{M} \rangle$, such that: (*i*) $\mathcal{C}$ is the data domain of $\mathcal{L}$ containing (OLAP) *data cells*, which are the basic SQL aggregations of $\mathcal{L}$ computed against the relational data source S alimenting $\mathcal{L}$; (*ii*) $\mathcal{J}$ is the set of *dimensions* of $\mathcal{L}$, i.e. the *functional attributes* (of S) with respect to which the underlying OLAP analysis is defined (in other words, $\mathcal{J}$ is the set of attributes with respect to which relational tuples in S are aggregated); (*iii*) $\mathcal{H}$ is the set of *hierarchies* related to the dimensions of $\mathcal{L}$, i.e. hierarchical representations of the functional attributes shaped-in-the-form-of generic trees; (*iv*) $\mathcal{M}$ is the set of *measures* of $\mathcal{L}$, i.e. the *attributes of interest* (of S) for the underlying OLAP analysis (in other words, $\mathcal{M}$ is the set of attributes with respect to which SQL aggregations stored in data cells of $\mathcal{L}$ are computed).

HISTOGRAMS

Histograms have been extensively studied and applied in the context of *Selectivity Estimation* (Kooi, 1980), and are effectively implemented in commercial systems (e.g., Oracle Database, IBM DB2 Universal Database, Microsoft SQL Server) to query optimization purposes. In statistical databases (Shoshani, 1997), histograms represent a method for approximating probability distributions. They have also been used in data mining activities, intrusion detection systems, scientific databases, i.e. in all those applications which (*i*) operate on huge numbers of detailed records, (*ii*) extract useful knowledge only from condensed information consisting of summary data, (*iii*) but are not usually concerned with detailed information. Indeed, histograms can reach a surprising efficiency and effectiveness in approximating the actual distributions of data starting from summarized information. This has led the research community to investigate the use of histograms in the fields of *DataBase Management Systems* (DBMS) (Kooi, 1980; Poosala & Ioannidis, 1997; Ioannidis & Poosala, 1999; Acharya, Poosala & Ramaswamy, 1999; Gunopulos, Kollios, Tsotras & Domeniconi, 2000; Bruno, Chaudhuri & Gravano, 2001), and *OnLine Analytical Processing* (OLAP) systems (Poosala & Ganti, 1999; Buccafurri, Furfaro, Saccà & Sirangelo, 2003; Cuzzocrea, 2005a; Cuzzocrea & Wang, 2007; Leng, Bao, Wang & Yu, 2007).

Histogram literature has a long history, what confirms the prominence of histogram research within the broader context of database/data-cube compression techniques. They are data structures obtained by partitioning a data distribution (or, equally, a data domain) into a number of mutually disjoint blocks, called *buckets*, and then storing, for each bucket, some aggregate information of the corresponding range of values, like the sum of values in that range (i.e., applying the SQL aggregate operator SUM), or the number of occurrences (i.e., applying the SQL aggregate operator COUNT), such that this information retains a certain "summarizing content".

Histograms are widely used to support two kinds of applications: (*i*) selectivity estimation inside *Query Optimizers* of DBMS, as highlighted before, and (*ii*) *approximate query answering* against databases and data cubes. In the former case, the data distribution to be compressed consists of the frequencies of values of attributes in a relation (it should be noted that, in this vest, histograms are mainly used within the core layer of DBMS, thus dealing with databases properly). In the latter case, the data distribution to be compressed consists of the data items of the target domain (i.e., a database or a data cube) directly, and the goal is to provide fast and approximate answers to resource-intensive queries instead of waiting-for time-consuming exact evaluations of queries. They are a very-popular class of synopsis data structures, so that they have been extensively used in the context of approximate query answering techniques. Early experiences concerning this utilization of histograms are represented by the work of Ioannidis and Poosala (1999), that propose using histograms to provide approximate answers to set-valued queries, and the work of Poosala and Ganti (1999), that propose using histograms to provide approximate answers to range-queries in OLAP.

There are several classes of histograms in literature. We distinguish between *one-dimensional histograms*, i.e. histograms devoted to compress one-dimensional data domains, and *multidimensional histograms*, i.e. histograms working on multidimensional data domains, which are more interesting than the former, and can be found in a wide range of modern, large-scale, *data-intensive* applications. Moreover, we can also further distinguish between *static histograms* and *dynamic histograms*. The first ones are statically computed against the target domain, and are not particularly suitable to efficiently accomplish data updates occurring on original data sources; the second ones are dynamically computed by taking into consideration, beyond the target domain, or, in some cases, a synopsis of it, other entities related to the dynamics of the target DBMS/OLAP server such as *query-workloads*, *query feedbacks*, *load balancing issues* etc. Contrarily to the previous histogram class, dynamic histograms efficiently support update management, being their partition dependent on a "parametric" configuration that can be easily (re-)computed at will.

With respect to our work, we are mainly interested in multidimensional histograms, since multidimensional data domains well-represent OLAP data cubes. A popular and conventional approach is based on the well-known *Attribute-Value Independence* (AVI) assumption, according to which any query involving a set of attributes can be answered by applying it on each attribute singularly. This approach is theoretically