

Providing Probabilistically-Bounded Approximate Answers to Non-Holistic Aggregate Range Queries in OLAP

Alfredo Cuzzocrea
DEIS Dept., University of Calabria
Via P. Bucci, 41C
87036 Cosenza, Italy
cuzzocrea@si.deis.unical.it

ABSTRACT

A novel framework for providing probabilistically-bounded approximate answers to non-holistic aggregate range queries in OLAP is presented in this paper. Such a framework allows us to efficiently support OLAP applications, as answering queries is the main bottleneck for this kind of applications. To this end, *scalability* of the techniques and *accuracy* of the answers are recognized as important limitations of state-of-the-art approximate query answering proposals in OLAP. Specifically, this paper is focused on the latter limitation, whereas it refers to results presented in [9] for the first one. The **KSyn** synopsis data structure, which implements the guidelines of the proposed framework and overcomes the recognized limitations, is also presented and discussed in detail, along with a query-conscious error metrics-based storage space allocation scheme. Finally, encouraging preliminary experimental results stating the goodness of our proposal are presented and discussed.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems – *Query Processing*.

General Terms

Algorithms, Performance.

Keywords

Approximate Query Answering, OLAP, Synopsis Data Structures, Error Guarantees, Probabilistic Models.

1. INTRODUCTION

OLAP is a successful technique for extracting summarized knowledge from very large data repositories stored in a Data Warehouse Server (DWS), thanks to the amenity of representing DW data according to a multidimensional and multi-resolution vision. As an example, in the *OLAP Mining* context [16], it is widely recognized that OLAP technology can add very useful features to the natural “explanatory” model [16] of the Data

Mining tasks inferring useful knowledge from huge amounts of data. In fact, OLAP technology efficiently supports the need for processing integrated views of data, and mining knowledge and patterns at different levels of abstraction via typical OLAP operations like drill-up, roll-down, pivot, slice and dice etc. In supporting such kind of applications, an *OLAP Engine* typically provides query answers and views, and makes available OLAP operations on massive multidimensional data. As a consequence, efficiently processing multidimensional data is the main bottleneck, and providing approximate answers to OLAP queries represents a very effective way of improving performances.

In [9], we recognize on two important limitations of approximate query answering techniques in OLAP: *scalability*, i.e. the capability of the techniques of scaling as input data cubes grow in size, and *accuracy*, i.e. the need for guarantees on the degree of approximation of the retrieved answers. Specifically, our techniques presented in [9] are tailored for *OLAP aggregate range queries* [18], a particular and very useful class of OLAP queries that are defined as the application of a SQL aggregation operator over a set of (given) contiguous ranges in the domains of the dimensions of the input data cube. Also in this paper we are interested in such kind of queries.

As regards the first issue, in [9] we propose a *dimensionality reduction* technique for multidimensional data cubes based on the well-known *Karhunen-Loeve Transform* (KLT) [21]. Furthermore, since KLT is a *lossy* transformation, so that it is not possible to go back to the original domain from the transformed domain without losing information, some effective optimizations have also been proposed [9] in order to tailor the KLT to the OLAP context. The first optimization consists in a MOLAP-based technique for representing input data cubes that allows us to significantly contrast the effect of the projection error due to the KLT [9]. The second one consists in devising a *query-conscious error metrics based technique* that allows us to make the KLT *robust* w.r.t. the approximation of the query answers, by selecting the *query-conscious optimal KLT output dimension number* [9].

As regards the specific need for theoretical guarantees on the degree of approximation of the answers, in this paper we propose a sampling-based technique that ensures (i) good quality of the approximate answers, and (ii) probabilistic guarantees on their degree of approximation. In more detail, the latter issue is faced off by building *statistical synopses* on the top of the input data cube *A* via sampling, and designing efficient query algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DOLAP'05, November 4–5, 2005, Bremen, Germany.

Copyright 2005 ACM 1-59593-162-7/05/0011...\$5.00.

running on them. *Synopsis data structures* are succinct representations of the input data cube, which are commonly built on the basis of a certain property inferred from the data distribution characterizing the data cube. In our proposal, the benefits due to sampling are: (i) to obtain the smallest computational complexity in processing multidimensional data among all the known-in-literature approximate query answering techniques, and (ii) to ensure probabilistic guarantees on the degree of approximation of the answers thanks to the underlying statistical framework we describe in this paper. Such a statistical framework is based on the *Hoeffding's inequality* [19], which is a well-known result coming from the theoretical statistics, belonging to the set of the so-called *tail inequalities*, which also comprises the *Markov's*, *Chebyshev's*, and *Chernoff's* ones. Particularly, the Hoeffding's inequality gives probabilistic bounds on the estimation of the average value of any instance of a given set of independent random variables. In other words, applying the Hoeffding's inequality allows us to obtain a *confidence interval* for the estimated parameter. In our framework, the independent random variables are built on the top of the sampled data cells belonging to the input data cube. Given a range query Q , the estimated parameter is thus equal to the value of the (approximate) AVG aggregate operator on the set of the involved-by- Q data cells (i.e., the involved instance of the set of random variables respectively). Starting from the AVG value, we easily derive the approximate answer for other popular SQL aggregation operators like SUM, COUNT etc.

Combining the two approaches (i.e., KLT and sampling) leads to the definition of the **KSyn**, a synopsis data structure that “codifies” our proposal. Given a multidimensional data cube A , the **KSyn**(A) data structure is built on the top of A by means of complex data processing algorithms that run in an *off-line mode*, as happens for usual maintenance operations of any modern OLAP server platform. The latter feature ensures that these algorithms are transparent for the user, and, above all, that (i) no computational overheads are introduced at query time, and (ii) query answering performances can be sensitively improved. At query time, each range query Q against A is intercepted and evaluated against **KSyn**(A), thus obtaining a fast and approximate answer to Q , denoted by $\tilde{A}(Q)$.

Obviously, the **KSyn** building task depends on the amount of storage space S available for housing it. Storing issues regarding the allocation of S for optimally housing the **KSyn** are discussed in Section 4.1, where we also present a *query-conscious error metrics-based storage space allocation scheme*, which is able to determine the *query-conscious optimal allocation scheme*, i.e. the allocation scheme that minimizes the query error.

We highlight that in both the critical aspects of our proposal (i.e., transforming issues, mostly discussed in [9], and storing issues, discussed in this paper) we adopt models and reasoning based on the analysis of a set of *synthetic queries* defined starting from the input data cube, and managed during the off-line processing phase. This is due to the fact that, being a-priori unknown both the nature of the query-workloads against the target data cube and the data distribution characterizing the same data cube, we can infer useful-for-processing knowledge by *mining the answers to synthetic queries only*. For this reason, we name our solutions as “query-conscious”.

The remaining part of the paper is organized as follows. In Section 2, we briefly outline related work on approximate query answering in OLAP. In Section 3, we present our problem statement and an overview on the proposed technique. In Section 4, we present an algorithm for efficiently building the **KSyn**, and investigate storing issues concerning the **KSyn** physical representation on secondary memory. In Section 5, we describe and discuss on the **KSyn** query model, focusing on the need for guarantees on the degree of approximation of the answers, from a theoretical point of view. In Section 6, we provide a preliminary experimental evaluation stating the goodness of our proposed technique. Finally, in Section 7, we draw the conclusions and indicate future work.

2. RELATED WORK

Due to relevant and growing interest from the academic and industrial worlds, a plethora of approximate query answering techniques have been proposed in these last years. Histograms are the most popular and investigated technique, mainly due to their capability of summarizing very large, multidimensional data cubes (or, equally, multidimensional data domains). A histogram represents a data cube as a collection of *buckets* (i.e., data blocks) that store a parameter summarizing the items contained in them (e.g., sum or average values). Since data cubes are usually enormous in size, in order to compute the histogram of a given data cube, greedy algorithms have been very often proposed. The criterion for generating buckets is a characteristic feature of each proposal. Some successful histograms proposed in literature are: *Equi-Depth* [22] *MHist* [24], *MinSkew* [2], *GenHist* [15], and *STHoles* [5] histograms. Scalability is the most important limitation for histograms, as they work very well on lowly dimensional data cubes, whereas they strongly suffer of reduced performances in representing highly dimensional data cubes, and, as a consequence, reduced capability of estimating queries over them.

Other proposals for approximate query answering fall in the more general class of the synopsis data structures. Among all, we cite: *Quasi-Cube* [4], which proposes building parametric log-linear models to obtain data cube compression, and *Indices* [6], which allow two-dimensional data cubes to be represented in very small portions of bytes. Just like histograms, both these techniques do not scale well on highly dimensional domains.

Wavelets [25,26] allow us to obtain compressed representations of data cubes via mathematical transformations that (i) can be implemented introducing low computational overheads, and (ii) scale better than histograms [25,26]. Nevertheless, wavelets are not useful on highly dimensional domains, as, in this specific case, due to mathematical issues, they present strong limitations in re-constructing the original domain from the transformed domain.

Sampling [1,3,7,14] is currently one of the most popular ways of supporting approximate query answering, mainly due to its low computational requirements. Sampling aims to obtain a small-in-size domain from a given massive domain, thus achieving a compressed representation of the latter. The major benefit carried by sampling is the amenity of processing (variants are in both online and off-line manners) sampled data by means of consolidated intelligent techniques, such as outlier detection and management [7]. For instance, Hellerstein *et al.* [17] propose a system for answering OLAP queries on ROLAP data cubes as to

obtain *online aggregation*. This system is based on sampling relational tuples and exploits the Hoeffding’s inequality for obtaining confidence intervals over the retrieved answers; in more detail, users can start the query evaluation via a GUI, and progressively enlarge the underlying sample set in order to achieve a better approximation of the retrieved answers. Therefore, our approach resembles that of Hellerstein *et al.* for what concerns the statistical framework. Nevertheless, there is a significant difference between the two approaches. In our framework, we codify the sampled data, and, in some sense, the same features of the Hoeffding’s inequality into the **KSyn** data structure in an *off-line* manner, and, then, we use the **KSyn** at query time to evaluate queries. Instead, Hellerstein *et al.* propose to answer queries in an *online* fashion, without storing any data structure. For this reason, we retain that the latter proposal is inadequate for the particular goal of querying (massive) MOLAP data cubes as drawn in our approach, being it, indeed, more suitable for online ROLAP data cubes.

However, as happens with the previous proposals, sampling presents important limitations when it is applied on highly dimensional domains, mainly for issues deriving from projection errors due to the same sampling phase.

To the best of our knowledge, even if accuracy of the answers has been already recognized as a fundamental research challenge for approximate query answering, in literature, there are not other proposals that, as the ours, aim to devise a formal statistical framework for ensuring theoretically-based guarantees on the degree of approximation of the answers. Another benefit deriving from our proposal is the amenity of using it in an integrated manner with state-of-the-art approximate query answering techniques in order to improve their capability of scaling on highly dimensional domains, and providing probabilistically-bounded approximate answers.

3. PROBLEM STATEMENT AND TECHNIQUE OVERVIEW

Let A be a data cube, let $D = \{d_0, d_1, \dots, d_{N-1}\}$ be the set of the dimensions of A , and let $M = \{m_0, m_1, \dots, m_{P-1}\}$ be the set of the measures of A . Our basic issue is providing an approximate answer $\tilde{A}(Q)$ to the following range query:

$$Q = Q^{AGG}(m_k) = AGG_{m_k}(l_{d_T} : u_{d_T}; l_{d_{T+1}} : u_{d_{T+1}}; \dots; l_{d_{T+V-1}} : u_{d_{T+V-1}}),$$

being: (i) $m_k \in M$ the measure of interest, such that $k \in \{0, 1, \dots, P-1\}$; (ii) AGG a non-holistic SQL aggregation operator [16] like AVG, SUM, COUNT etc; (iii) $\{d_T, d_{T+1}, \dots, d_{T+V-1}\}$ the set of the V dimensions involved by Q , such that $1 \leq V \leq N$; and (iv) $[l_{d_i} : u_{d_i}]$ the range on the dimension d_i , such that $i \in \{0, 1, \dots, N-1\}$ (in particular, l_{d_i} and u_{d_i} are the lower bound and the upper bound of the range respectively).

In our framework, we define the *Approximation Error* $\Sigma(Q)$ as the difference between the exact answer to Q , denoted by $A(Q)$, which is evaluated on A , and the approximate answer to Q , $\tilde{A}(Q)$, which is evaluated on **KSyn**(A). Thus, $\Sigma(Q) = A(Q) - \tilde{A}(Q)$. $\Sigma(Q)$ is composed by two amounts of error: (i) the *Transformation Error* $\Sigma_K(A)$ introduced by the KLT, and (ii) the *Statistical Error* $\Sigma_S(A, Q)$ caused by the sampling. Formally, $\Sigma(Q) = \Sigma_K(A) +$

$\Sigma_S(A, Q)$. We highlight that, thanks to the our statistical framework, which is presented in Section 5, $\Sigma_S(A, Q)$ is low, so that $\Sigma(Q) \approx \Sigma_K(A)$, i.e. $\Sigma(Q)$ is mainly due to the KLT. Therefore, it follows that minimizing $\Sigma_K(A)$ is a critical requirement for the effectiveness of our proposed technique. KLT-focused optimizations presented in [9] have been designed to satisfy this proper, particular constraint. We remind the reader to [9] for further details about these optimizations.

To better understand the remaining part of the paper, here we present a summary about the whole technique we propose, by also highlighting the results that have been provided in [9]. Given the input data cube A , the **KSyn**(A) building task can be summarized in the following steps:

- Compute the MOLAP-based representation of A , said $Y_{MOLAP}(A)$, as presented in [9].
- For each data cube y_m in $Y_{MOLAP}(A)$, obtain the KLT-based transformed data cube, denoted by $KLT(y_m)$, according to the our query-conscious error metrics-based technique (which, as said above, is an optimization for the KLT [9]). This originates the $Y_{MOLAP}^T(A)$ set, which is composed by the (MOLAP) M_{i, δ_i}^{T, n_i} data cubes (particularly, n_i denotes the number of dimensions and δ_i denotes the cardinality of each dimension of the data cube respectively – see [9] for further details), as presented in [9].
- Compute the allocation of the available storage space S across the $Y_{MOLAP}^T(A)$ set, according to the our query-conscious error metrics-based allocation scheme.
- Uniformly sample each data cube M_{i, δ_i}^{T, n_i} in $Y_{MOLAP}^T(A)$, thus obtaining a *sampled data block*, denoted by SDB_i , which is the elementary component of the **KSyn**(A) data structure.

It should be noted that whatever is the storage model of the input data cube A (it can be ROLAP, MOLAP, or HOLAP), our representation technique allows us to obtain MOLAP data cubes, which can also be intended as a materialized, fully aggregated data structure. Thus, in the rest of the paper we assume of dealing with such kind of data cubes.

4. BUILDING AND STORING **KSyn**

4.1 Preliminaries and Definitions

Given the $Y_{MOLAP}^T(A)$ set and the storage space S available for housing the **KSyn**(A) data structure, the problem addressed by the storage space allocation scheme is to determine a *proportional allocation* of S across the $Y_{MOLAP}^T(A)$ set, i.e. to find, for each M_{i, δ_i}^{T, n_i} , the portion of S , namely $S(M_{i, \delta_i}^{T, n_i})$, composed by (i) the amount needed for housing the samples extracted from M_{i, δ_i}^{T, n_i} , and (ii) the amount needed for housing the multidimensional region of M_{i, δ_i}^{T, n_i} w.r.t. the A multidimensional space (i.e., by storing the original bounds of the region), namely $R(M_{i, \delta_i}^{T, n_i})$. As we better explain in Section 5, the $R(M_{i, \delta_i}^{T, n_i})$ multidimensional region is used to support query evaluation on **KSyn**(A). We

denote the amount of $S(M_{i,\delta_i}^{T,n_i})$ spent for housing the samples as $S_{eff}(M_{i,\delta_i}^{T,n_i})$, and the amount of $S(M_{i,\delta_i}^{T,n_i})$ spent for housing $R(M_{i,\delta_i}^{T,n_i})$ as $E(M_{i,\delta_i}^{T,n_i})$. Thus, $S(M_{i,\delta_i}^{T,n_i}) = S_{eff}(M_{i,\delta_i}^{T,n_i}) + E(M_{i,\delta_i}^{T,n_i})$. We highlight that $R(M_{i,\delta_i}^{T,n_i})$ can be efficiently represented (i.e., investing a bit of space) on memory by storing, for each $d_{i,j}^T$ dimension of M_{i,δ_i}^{T,n_i} , such that $j \in \{0, 1, \dots, n_i-1\}$, the lower bound l_i^T and the upper bound u_i^T of $d_{i,j}^T$ w.r.t. the A multidimensional space, so that, usually, $E(M_{i,\delta_i}^{T,n_i}) \ll S_{eff}(M_{i,\delta_i}^{T,n_i})$ and $S(M_{i,\delta_i}^{T,n_i}) \approx S_{eff}(M_{i,\delta_i}^{T,n_i})$.

In our framework, the allocation of S across the $Y_{MOLAP}^T(A)$ set is provided by a query-conscious error metrics-based storage allocation scheme, denoted by AS .

Given the input data cube A , $\mathbf{KSyn}(A)$ is composed by B uniformly sampled data blocks SDB_i , such that $i \in \{0, 1, \dots, B-1\}$, plus an indexing data structure, called $\mathbf{K-Index}$, which allows the $\mathbf{KSyn}(A)$ to be efficiently accessed at query time (particularly, the $\mathbf{K-Index}$ stores the $R(M_{i,\delta_i}^{T,n_i})$ multidimensional regions). Each SDB_i is built by sampling each corresponding M_{i,δ_i}^{T,n_i} data cube.

Sampling depends on the amount of storage space available for housing the sampled data, and it is widely recognized that the bigger is such amount, the more is *accurate* the sampling. In our framework, this amount of storage space is determined by the allocation scheme AS .

In order to present the AS scheme, here we introduce the definition of *volume* of a data cube. Letting M be a n -dimensional MOLAP data cube, the volume of M , denoted by $\|M\|$, is defined as the product of the cardinalities of the dimensions of M : $\|M\| = \prod_{i=0}^{n-1} |d_i|$. Analogously, we can extend this definition as to

include the volume of a range query Q (i.e., $\|Q\|$), and the volume of a multidimensional region R (i.e., $\|R\|$).

4.2 Query-Conscious Error Metrics-based Storage Space Allocation Scheme

The AS scheme we propose is based on the criterion of proportionally assigning more storage space to those data cubes that present (i) a high value of volume, and (ii) skewed characteristic data distribution, and, contrarily, proportionally assigning less storage space to those data cubes that present low values of the previous (i) and (ii) parameters. In the AS scheme, the constraint to be satisfied is that the final (space) allocation must be contained within S :

$$\sum_{i=0}^{B-1} S(M_{i,\delta_i}^{T,n_i}) \leq S \quad (1)$$

The ratio of such a criterion is the following: we assign more storage space to those data cubes having a high value of volume (note that this can be generally considered as an indicator for the size of the data cube) and skewed data distribution as these two

conditions make them requiring a larger sample set than data cubes non-adhering to these conditions. As we better explain in Section 5, when we deal with the former kind of data cubes, a large sampling *gives the possibility* to obtain an acceptable quality of the degree of approximation of the answers.

While determining the volume of a data cube is trivial, deciding if the data distribution of a data cube is uniform or skewed is a well-known-in-literature problem. To solve this issue, we propose a model based on a query-conscious error metrics (this model can also be intended as a *heuristic*).

Given a M_{i,δ_i}^{T,n_i} data cube belonging to the $Y_{MOLAP}^T(A)$ set, in order to determine the nature of the data distribution of M_{i,δ_i}^{T,n_i} , we initially assume that data in M_{i,δ_i}^{T,n_i} are uniformly distributed, i.e. the *Continuous Value Assumption* (CVA) [8] holds. Under the CVA, letting Q^{SUM} be a range-SUM query, and $SUM(M_{i,\delta_i}^{T,n_i})$ be the summation of all the data cells of M_{i,δ_i}^{T,n_i} , a well-known-in-literature result [6,8] claims that the approximate answer to Q^{SUM} , said $\tilde{A}(Q^{SUM})$, can be evaluated as follows:

$$\tilde{A}(Q^{SUM}) = \frac{\|Q^{SUM} \cap M_{i,\delta_i}^{T,n_i}\|}{\|M_{i,\delta_i}^{T,n_i}\|} \cdot SUM(M_{i,\delta_i}^{T,n_i}) \quad (2)$$

i.e. by using linear interpolation. Also, as in [9], letting $Q_S^*(M_{i,\delta_i}^{T,n_i}) = \{q_0, q_1, \dots, q_{NQ^*(M_{i,\delta_i}^{T,n_i})-1}\}$ be the set of *all* the queries that can be defined on M_{i,δ_i}^{T,n_i} by considering, for each dimension d_i in M_{i,δ_i}^{T,n_i} , *all* the ranges having size that varies from 1 to $|d_i|$, and letting $NQ^*(M_{i,\delta_i}^{T,n_i})$ be the number of such queries (note that $NQ^*(M_{i,\delta_i}^{T,n_i})$ is a finite quantity), we introduce the $\mathcal{E}_{CVA}^2(M_{i,\delta_i}^{T,n_i})$ quadratic error as follows:

$$\mathcal{E}_{CVA}^2(M_{i,\delta_i}^{T,n_i}) = \frac{1}{NQ^*(M_{i,\delta_i}^{T,n_i})} \cdot \sum_{k=0}^{NQ^*(M_{i,\delta_i}^{T,n_i})-1} (A(q_k) - \tilde{A}_{CVA}(q_k))^2 \quad (3)$$

where:

$$\tilde{A}_{CVA}(q_k) = \frac{\|q_k \cap M_{i,\delta_i}^{T,n_i}\|}{\|M_{i,\delta_i}^{T,n_i}\|} \cdot SUM(M_{i,\delta_i}^{T,n_i}) \quad (4)$$

We highlight that, as motivated in [9], we are considering range-SUM queries as their aggregation operator (i.e., SUM) is such that they pick, in some sense, the contributions of *all* the involved data cells (i.e., range-SUM queries are “additive” queries by definition – see [9] for further details).

The second step is assuming that data in M_{i,δ_i}^{T,n_i} are skewed. Under this assumption, we build the *Zipf version* of M_{i,δ_i}^{T,n_i} , denoted by M_{i,δ_i}^{Z,n_i} , as follows. Starting from M_{i,δ_i}^{T,n_i} , we initially obtain a Zipf data distribution with parameter z set to 1.0, namely $Z^{1.0}(M_{i,\delta_i}^{T,n_i})$, setting as input generating set Ψ (i.e., the set composed by values that will appear as points in the Zipf distribution) the set

composed by the data cells C_k of M_{i,δ_i}^{T,n_i} , enriched with other information. In more detail, Ψ is populated with $O_k = \langle C_k, R_k \rangle$ objects, such that C_k is a data cell of M_{i,δ_i}^{T,n_i} , and R_k is its multidimensional reference. As a consequence, each point in $Z^{1.0}(M_{i,\delta_i}^{T,n_i})$ is also an object $P_k = \langle C_k, R_k \rangle$ such that C_k is a data cell extracted from M_{i,δ_i}^{T,n_i} according to the 1.0-Zipf distribution, and R_k is its multidimensional reference which is held during the extraction. Note that $z = 1.0$ represents the ‘‘border condition’’ between the so-called *lowly-skewed Zipf distributions* (i.e., Zipf distributions for which the z parameter is such that $0 < z \leq 0.9$) and the so-called *highly-skewed Zipf distributions* (i.e., Zipf distributions for which the z parameter is such that $z > 1.0$). After that, we build the M_{i,δ_i}^{Z,n_i} data cube by loading the value of each object P_k of $Z^{1.0}(M_{i,\delta_i}^{T,n_i})$ (i.e., C_k), and setting it in the M_{i,δ_i}^{Z,n_i} multidimensional space on the basis of the corresponding multidimensional reference R_k . Finally, analogously to the previous case, we introduce the $\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$ quadratic error as follows:

$$\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i}) = \frac{1}{NQ^s(M_{i,\delta_i}^{T,n_i})} \cdot \sum_{k=0}^{NQ^s(M_{i,\delta_i}^{T,n_i})-1} (A(q_k) - \tilde{A}_{ZIPF}(q_k))^2 \quad (5)$$

where $\tilde{A}_{ZIPF}(q_k)$ is the approximate answer to q_k evaluated on the M_{i,δ_i}^{Z,n_i} data cube. While the ratio underlying the definition of the $\varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i})$ quadratic error is clear, we highlight that the meaning of the $\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$ quadratic error is driven by the following steps: (i) assume that M_{i,δ_i}^{T,n_i} is skewed, (ii) generate its Zipf version (i.e., M_{i,δ_i}^{Z,n_i}), and (iii) measure how much the answers to queries evaluated on M_{i,δ_i}^{Z,n_i} are ‘‘distant’’ from the answers to the same queries evaluated on M_{i,δ_i}^{T,n_i} . Therefore, $\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$ is an indicator for measuring how much data in M_{i,δ_i}^{T,n_i} are skewed.

After this pre-processing phase, we decide on the nature of M_{i,δ_i}^{T,n_i} by comparing $\varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i})$ and $\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$: if $\varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i}) \leq \varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$, then we can claim that, *with high probability*¹, M_{i,δ_i}^{T,n_i} data are uniform; otherwise (i.e., $\varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i}) > \varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$), we can claim that, *with high probability*, M_{i,δ_i}^{T,n_i} data are skewed. This approach is codified by the following *Deciding Function* $u(M_{i,\delta_i}^{T,n_i})$:

$$u(M_{i,\delta_i}^{T,n_i}) = \begin{cases} 0 & \text{if } \varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i}) \leq \varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i}) \\ 1 & \text{if } \varepsilon_{CVA}^2(M_{i,\delta_i}^{T,n_i}) > \varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i}) \end{cases} \quad (6)$$

¹ The reader should keep in mind that we are using a heuristic.

that, given a MOLAP data cube M , returns 0 if data in M are uniform, or 1 if data in M are skewed. We highlight that in (6) we are assuming a binary output. A more refined approach could be devised by proposing a finer analysis according to which, given a data distribution f and its domain ϕ , we can identify sub-ranges of ϕ in which f is uniform, sub-ranges of ϕ in which f is skewed, and, finally, sub-ranges of ϕ in which f presents intermediate features of both the two opposite situations.

Finally, we provide the allocation of S across the $Y_{MOLAP}^T(A)$ set according to the following proportional scheme AS :

$$\begin{cases} S(M_{i,\delta_i}^{T,n_i}) = S_{eff}(M_{i,\delta_i}^{T,n_i}) + E(M_{i,\delta_i}^{T,n_i}) \\ \sum_{i=0}^{B-1} S(M_{i,\delta_i}^{T,n_i}) \leq S \end{cases} \quad (7)$$

where:

$$S_{eff}(M_{i,\delta_i}^{T,n_i}) = \left[\left(\frac{\|M_{i,\delta_i}^{T,n_i}\|}{\sum_{k=0}^{B-1} \|M_{k,\delta_k}^{T,n_k}\|} + u(M_{i,\delta_i}^{T,n_i}) \cdot \frac{\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})}{\sum_{k=0}^{B-1} u(M_{k,\delta_k}^{T,n_k}) \varepsilon_{ZIPF}^2(M_{k,\delta_k}^{T,n_k})} \right) \cdot S \right] \quad (8)$$

Since the meaning of the $E(M_{i,\delta_i}^{T,n_i})$ is trivial, the ratio of the scheme (7) is mainly given by the relation (8). The first term (i.e., $\frac{\|M_{i,\delta_i}^{T,n_i}\|}{\sum_{k=0}^{B-1} \|M_{k,\delta_k}^{T,n_k}\|}$) allocates a proportionally greater amount of space to the M_{i,δ_i}^{T,n_i} data cube as greater is its volume compared with the volumes of the other M_{k,δ_k}^{T,n_k} data cubes. The second term (i.e., $u(M_{i,\delta_i}^{T,n_i}) \cdot \frac{\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})}{\sum_{k=0}^{B-1} u(M_{k,\delta_k}^{T,n_k}) \varepsilon_{ZIPF}^2(M_{k,\delta_k}^{T,n_k})}$) provides a non-null contribution when data in M_{i,δ_i}^{T,n_i} are skewed (i.e., $u(M_{i,\delta_i}^{T,n_i}) = 1$). In this latter case, it allocates a proportionally greater amount of space to M_{i,δ_i}^{T,n_i} as greater is its $\varepsilon_{ZIPF}^2(M_{i,\delta_i}^{T,n_i})$ quadratic error compared with the quadratic errors introduced by the other M_{k,δ_k}^{T,n_k} skewed data cubes. Note that, when data in M_{i,δ_i}^{T,n_i} are uniform (i.e., $u(M_{i,\delta_i}^{T,n_i}) = 0$), the scheme (7) allocates the storage space to M_{i,δ_i}^{T,n_i} on the basis of its volume only. We highlight that this is well-founded in presence of uniform data, as the skewness of the data cube is not considered in this case.

4.3 The allocateStorageSpace Algorithm

The `allocateStorageSpace` algorithm (see Figure 1) takes as input the $Y_{MOLAP}^T(A)$ set and the amount of storage space S , and returns the allocation of S across $Y_{MOLAP}^T(A)$ on the basis of the scheme (7). In particular, the final allocation scheme is modeled as an array containing in each cell i the proportional amount of S to be allocated for each M_{i,δ_i}^{T,n_i} data cube.

Algorithm allocateStorageSpace

Input: The $Y_{MOLAP}^T(A)$ set, and the amount of storage space S .

Output: The allocation of S across $Y_{MOLAP}^T(A)$.

Begin

```

TDCNumber  $\leftarrow$   $Y_{MOLAP}^T.size$ ;
volArray  $\leftarrow$  initializeVolArray(TDCNumber);
epsZArray  $\leftarrow$  initializeEpsZArray(TDCNumber);
decFuncArray  $\leftarrow$  initializeDecFuncArray(TDCNumber);
regStoArray  $\leftarrow$  initializeRegStoArray(TDCNumber);
allocArray  $\leftarrow$  initializeAllocArray(TDCNumber);
i  $\leftarrow$  0;
while (i < TDCNumber) do
   $M_i^T \leftarrow Y_{MOLAP}^T.get(i)$ ;
  vol  $\leftarrow$  computeVolume( $M_i^T$ );
  add(vol, volArray);
   $NQ^* \leftarrow$  computeMaxQueryNumber( $M_i^T$ );
   $Q_S \leftarrow$  buildTrainingQuerySet( $M_i^T$ );
  epsCVA  $\leftarrow$  computeCVAQuadError( $M_i^T, Q_S$ );
  epsZIPF  $\leftarrow$  computeZIPFQuadError( $M_i^T, Q_S$ );
  add(epsZIPF, epsZArray);
  decFunc  $\leftarrow$  computeDecFunc(epsCVA, epsZIPF);
  add(decFunc, decFuncArray);
  regStorage  $\leftarrow$  computeRegStorage( $M_i^T$ );
  add(regStorage, regStoArray);
  i++;
end;
allocArray  $\leftarrow$  computeAlloc(volArray, decFuncArray,
                           epsZArray, regStoArray, S);
return allocArray;

```

End;

Figure 1. The allocateStorageSpace algorithm.

We highlight that, in the allocateStorageSpace algorithm, without loss of generality, we implicitly are assuming that the constraint of the scheme (7) is always satisfied. Indeed, we also developed a more robust version of the allocateStorageSpace algorithm, called allocateStorageSpace⁺, which dynamically checks the constraint during the execution. If the constraint is violated and some M_{i,δ_i}^{T,n_i} data cubes for which an amount of storage space must be allocated exist, the allocateStorageSpace⁺ algorithm obtains new free space to be invested for the latter data cubes by (i) sorting the data cubes for which the allocation was completed by descending size of the (allocated) storage space, and (ii) proportionally removing from the allocations of the first Z sorted data cubes the needed storage space, by further applying the scheme (7) on these Z data cubes only (in particular, note that Z is a parameter that depends on the needed storage space). For space reasons, here we do not present the allocateStorageSpace⁺ algorithm.

4.4 The buildKSyn Algorithm, and the KSyn Physical Representation

In this Section, we provide a brief overview on (i) the KSyn building task, and (ii) the KSyn physical representation on secondary memory.

Given the $Y_{MOLAP}^T(A)$ set and the allocation scheme AS , computed through the allocateStorageSpace algorithm (see Figure 1), the KSyn(A) building task generates, for each $i \in \{0, 1, \dots, B-1\}$,

the SDB_i sampled data block from the M_{i,δ_i}^{T,n_i} data cube by taking as physical bound for housing the sampled data the amount of storage space $S_{eff}(M_{i,\delta_i}^{T,n_i})$ (i.e., $SIZE(SDB_i) \leq S_{eff}(M_{i,\delta_i}^{T,n_i})$). At the end of this process, KSyn(A) is composed by all the generated SDB_i . Note that, letting s be the disk occupation of a sample on the target host, the max number of samples that can be extracted from M_{i,δ_i}^{T,n_i} within $S_{eff}(M_{i,\delta_i}^{T,n_i})$ is $\left\lfloor \frac{S_{eff}(M_{i,\delta_i}^{T,n_i})}{s} \right\rfloor$. The **K-Index**

indexing data structure (see Section 4.1) is dynamically built along with the KSyn(A) data structure. Each entry of **K-Index** indexes a SDB_i by storing its $R(M_{i,\delta_i}^{T,n_i})$ multidimensional region as a collection of $\langle l_i^T, u_i^T \rangle$ couples for each dimension $d_{i,j}^T$ of the corresponding M_{i,δ_i}^{T,n_i} data cube (recall that SDB_i are extracted from M_{i,δ_i}^{T,n_i}), such that l_i^T and u_i^T are the lower bound and the upper bound of $d_{i,j}^T$ respectively. As stated in Section 4.1, $R(M_{i,\delta_i}^{T,n_i})$ occupies an amount of storage space equal to $E(M_{i,\delta_i}^{T,n_i})$.

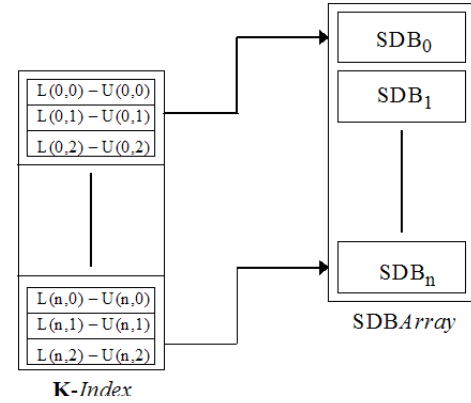


Figure 2. KSyn physical representation on secondary memory.

The KSyn(A) data structure is thus represented on secondary memory as an array of sampled data blocks, called SDBArray, indexed by the **K-Index** (see Figure 2).

The KSyn(A) building task is codified by the buildKSyn algorithm (see Figure 3).

Algorithm buildKSyn

Input: The $Y_{MOLAP}^T(A)$ set, and the allocation scheme AS .

Output: Void (it builds KSyn(A) and stores it on disk).

Begin

```

KSyn  $\leftarrow$   $\emptyset$ ;
TDCNumber  $\leftarrow$   $Y_{MOLAP}^T.size$ ;
SDBArray  $\leftarrow$  initializeSDBArray(TDCNumber);
KIndex  $\leftarrow$  initializeKIndex(TDCNumber);
i  $\leftarrow$  0;
while (i < TDCNumber) do
   $M_i^T \leftarrow Y_{MOLAP}^T.get(i)$ ;
   $S_i \leftarrow AS.getSampleAlloc(i)$ ;
   $SDB_i \leftarrow sample(M_i^T, S_i)$ ;
  add( $SDB_i, SDBArray$ );

```

```

     $E_i \leftarrow AS.getMDRegAlloc(i);$ 
     $I_i \leftarrow computeIndexEntry(R(M_i^T), E_i);$ 
     $add(I_i, KIndex);$ 
     $setReference(I_i, SDB_i);$ 
     $i++;$ 
end;
KSyn  $\leftarrow create(SDBArray, KIndex);$ 
storeOnDisk(KSyn);
End;
```

Figure 3. The buildKSyn algorithm.

Data and update management in the *SDBArray* and the maintenance of the *K-Index* are important research challenges for our proposal, but they are outside of the scope of the paper, so that they must be considered as future work.

5. PROVIDING PROBABILISTIC BOUNDS ON THE APPROXIMATE ANSWERS

In this Section, we present our statistical framework for providing probabilistic guarantees on the degree of approximation of the answers, which, as discussed in Section 1, *should* be the key feature of any approximate query answering technique.

Let Q be a range query, and SDB_i be a sampled data block in the *SDBArray* such that the corresponding multidimensional region $R(M_{i,\delta_i}^{T,n_i})$, mapped on an entry of the *K-Index*, has a non-null intersection with Q (i.e., $Q \cap R(M_{i,\delta_i}^{T,n_i}) \neq \emptyset$). Here, we investigate and discuss about the specific issue of evaluating Q on SDB_i , in order to provide an approximate answer to Q , $\tilde{A}(Q)$, and a *reasonably* small confidence interval for it, denoted by $I(Q)$. As discussed in Section 1, in our proposed framework, $I(Q)$ is also rigorously theoretically-based.

A well-known-in-literature assumption that gives theoretical bases to the our approach states that using a tail inequality (like the Hoeffding's one) to *estimate* a given observed parameter p on a *independent* random variable set H (or, equally, to obtain a confidence interval for p such that $p = \tilde{p} \pm \Delta p$, where \tilde{p} is the estimation of p , and $2 \cdot \Delta p$ is the size of the confidence interval), one commits, *with high probability*, an (estimation) error that is at most ϵ_p with probability at least $1 - \Delta p$, being (i) ϵ_p a positive integer, and (ii) Δp the deviation from the exact value of p .

In more detail, in our framework, we use the Hoeffding's inequality for building the *KSyn* in a similar way to the method proposed by Hellerstein *et al.* for obtaining online aggregation [17]. The Hoeffding's inequality asserts the following [19]. Let $H = \{H_0, H_1, \dots, H_{L-1}\}$ be a set of independent random variables, r be a scalar, such that $0 \leq H_l \leq r$ for each $l \in \{0, 1, \dots, L-1\}$, $\bar{H} = \frac{1}{L} \sum_{l=0}^{L-1} H_l$ be the sample mean of the H set, and μ be the average value of *any instance* of the H set. Then, for each $\epsilon > 0$:

$$P(|\bar{H} - \mu| \leq \epsilon) \geq 1 - 2e^{-\frac{2L\epsilon^2}{r^2}} \quad (9)$$

Therefore, we obtain a probabilistic bound for the event $\mu = \bar{H} \pm \epsilon$, i.e. a probabilistic bound for the estimation of μ .

Furthermore, letting P_x be a *fixed* probability value, we can obtain the corresponding error value ϵ_x according to (9) as follows:

$$\epsilon_x = r \cdot \left(\frac{1}{2L} \ln \left(\frac{2}{1 - P_x} \right) \right)^{\frac{1}{2}} \quad (10)$$

We highlight that the latter property (i.e., computing a confidence interval for a given probability value) could be very useful to design and develop an *Approximate Query Engine Server* (AQES) supporting *query protocols* that allow OLAP users/applications and the target DWS to *mediate* on the degree of approximation of the (retrieved) answers, in a similar way to QoS-based network protocols. This leads to the definition of new paradigms for next-generation Data Warehouses applications.

In order to usefully adopt the Hoeffding's inequality inside our framework, a *probabilistic model* must be introduced. To this end, here we need to determine an independent random variable set, namely H , and what *and instance* of this set, namely G , is. Therefore, given a M_{i,δ_i}^{T,n_i} data cube, we define $H(M_{i,\delta_i}^{T,n_i})$ as the set composed by random variables $H_l(M_{i,\delta_i}^{T,n_i}): D \rightarrow M_{i,\delta_i}^{T,n_i}$, where D is the dimension set of M_{i,δ_i}^{T,n_i} , such that each $H_l(M_{i,\delta_i}^{T,n_i})$ takes a *uniformly sampled* multidimensional reference $\mathbf{p} = \langle p_0, p_1, \dots, p_{N-1} \rangle$ in the D space, and returns a data cell C in M_{i,δ_i}^{T,n_i} (i.e., $H_l(M_{i,\delta_i}^{T,n_i})(\mathbf{p}) = M_{i,\delta_i}^{T,n_i}[p_0][p_1] \dots [p_{N-1}] = C$). In other words, *the instance of the random variable set $H(M_{i,\delta_i}^{T,n_i})$ is just the sampled data block SDB_i , extracted from the M_{i,δ_i}^{T,n_i} data cube.* We recall that the number of samples that can be uniformly extracted from M_{i,δ_i}^{T,n_i} , according to the our allocation scheme (7), is bounded by the $\left\lfloor \frac{S_{eff}(M_{i,\delta_i}^{T,n_i})}{s} \right\rfloor$ quantity (see Section 4.2). Therefore, the instance of $H(M_{i,\delta_i}^{T,n_i})$, which we denote as

$$G(M_{i,\delta_i}^{T,n_i}) = \left\{ g_0, g_1, \dots, g_{\left\lfloor \frac{S_{eff}(M_{i,\delta_i}^{T,n_i})}{s} \right\rfloor - 1} \right\},$$

is formed by the data items contained in SDB_i , i.e. it is equal to a sub-set of data cells in M_{i,δ_i}^{T,n_i} . Obviously, this approach defined for one M_{i,δ_i}^{T,n_i} data cube must be extended to all the items belonging to the $Y_{MOLAP}^T(A)$ set (see Section 3), so that we totally obtain a set of random variables $H(M_{i,\delta_i}^{T,n_i})$ for each M_{i,δ_i}^{T,n_i} , or, more properly, a *multivariate stochastic system* $\Phi = \{H(M_{0,\delta_0}^{T,n_0}), H(M_{1,\delta_1}^{T,n_1}), \dots, H(M_{J-1,\delta_{J-1}}^{T,n_{J-1}})\}$, such that $J = |Y_{MOLAP}^T(A)|$.

Note that the uniform sampling ensures that the random variables are independent, as required by the Hoeffding's inequality [19].

Tailoring these theoretical results to efficiently support approximate query answering in OLAP is a critical aspect of our work. Consider, for simplicity, a one-dimensional data cube A and a query $Q = \text{AVG}(a, b)$ against it, such that $[a, b]$ is the query range. For the sake of simplicity, assume that A can be represented

with only one mono-dimensional MOLAP data cube M (see Section 3). To obtain a probabilistically-bounded answer to Q , $\tilde{A}(Q)$, we exploit the relation (9) and apply the definition of sample mean to the random variable set $H(M_1^T)$ by considering the instance of it built on the top of the data items in SDB_I involved by Q , said $G_{[a,b]}(M_1^T)$:

$$\tilde{A}(Q) = \bar{H}(M_1^T) \pm \varepsilon_Q = \frac{1}{|G_{[a,b]}(M_1^T)|} \cdot \sum_{k=0}^{|G_{[a,b]}(M_1^T)|-1} g_k \pm \varepsilon_Q \quad (11)$$

and committing an error ε_Q equal to (from (10)):

$$\varepsilon_Q = (b-a) \cdot \left(\frac{1}{2 \cdot |G_{[a,b]}(M_1^T)|} \ln \left(\frac{2}{1-P_Q} \right) \right)^{\frac{1}{2}} \quad (12)$$

where P_Q is an *unknown* probability value, but *probabilistically-bounded* as follows (from (10)):

$$P_Q \geq 1 - 2e^{-\frac{2 \cdot |G_{[a,b]}(M_1^T)| \cdot (\varepsilon_Q)^2}{(b-a)^2}} \quad (13)$$

thus obtaining probabilistic guarantees on $\tilde{A}(Q)$. From (13), note that the bigger is the size of the query range (i.e., $|b-a|$), the greater is P_Q : in other words, for “large” queries we obtain *better approximations*, i.e. approximations with larger confidence intervals, rather than for “small” queries, as expected.

Extending the above described methodology for dealing with multidimensional data cubes and queries defined on the top of other non-holistic SQL aggregation operators is trivial.

6. PRELIMINARY EXPERIMENTS

Preliminary experiments stating the goodness of the **KSyn** in supporting approximate query answering in OLAP are presented in this Section.

In these experiments, we defined the $A_Z(Z_l, Z_u, n, s)$ *Zipf-aware synthetic data cube class*, whose data cells are generated by a Zipf data distribution. In such data cubes, (i) $[Z_l, Z_u]$, such that $Z_l < Z_u$, is the range from which the z parameter of the Zipf generating distribution is uniformly extracted, (ii) n is the number of dimensions, and (iii) s is the *sparseness coefficient*. We highlight that the sparseness coefficient is a critical parameter in modeling synthetic data cubes, which cannot be ignored, as real-life data cubes are usually very sparse. Typically, in real-life data cubes, s ranges between 0.0001% and 1% respect to the total number of data cells.

This approach in the experimental validation of the proposed technique was due to the fact that our previous experiences concerning synthetic data cubes [10,11] has persuaded us that the Zipf one is the most reliable “inside-the-laboratory” distribution. In other words, the Zipf distribution models real-life phenomena most closely than other data distributions. On the other hand, this evidence has been already recognized by several previous similar experiences, like [23].

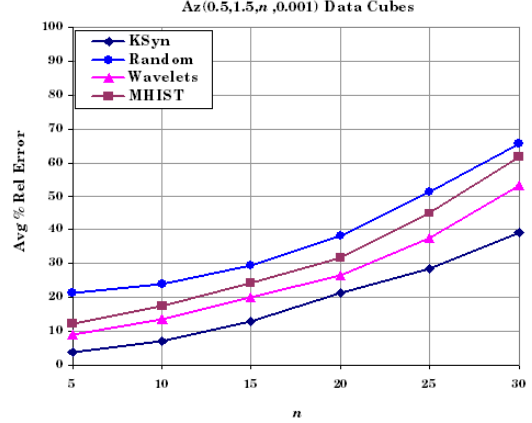


Figure 4. Variation of the average percentage relative error w.r.t. the dimension number.

Having defined the data cube class, we built the $A_Z(0.5, 1.5, n, 0.001)$ data cube instance, where n was posed to be a free parameter. As a consequence, the size of $A_Z(0.5, 1.5, n, 0.001)$ depends on n ; for instance, when $n = 10$, the $A_Z(0.5, 1.5, n, 0.001)$ data cube contained about 5.3×10^6 non-zero data cells, which, in our hardware infrastructure supporting the experimental framework, corresponded to an occupation of around 1.25 GB disk space. Similarly to [9], we also conducted some experiments by ranging the *selectivity* of the queries, which is another relevant parameter to be considered during the modeling of such an experimental framework. To this end, we defined a set of input (synthetic) range-SUM queries $Q_S(P)$ whose range sizes depend on a given integer parameter P , called *Range Increase*; note that P is defined in such a way that $Q_S(P)$ always contains increasing-inselectivity queries (see [9] for further details). By considering different P_j parameters, we obtained several $Q_S(P_j)$ query sets, whose total number is denoted by $N_T(Q_S)$. Finally, to test the quality of our proposed technique, we measured the average percentage relative error \bar{e}_{rel} , defined as follows:

$$\bar{e}_{rel} = \frac{1}{N_T(Q_S)} \cdot \sum_{k=0}^{N_T(Q_S)} \bar{E}_{rel}(P_k) \quad (14)$$

where $\bar{E}_{rel}(P_k)$ is the average percentage relative error measured taking as input the $Q_S(P_k)$ query set. In turn, $\bar{E}_{rel}(P_k)$ is obtained on the top of the percentage relative errors $\varepsilon_{rel}(Q_j)$ measured on each query Q_j belonging to the $Q_S(P_k)$ query set. $\varepsilon_{rel}(Q_j)$ is defined as follows:

$$\varepsilon_{rel}(Q_j) = \frac{|A(Q_j) - \tilde{A}(Q_j)|}{\max\{sb(I_{DC}), A(Q_j)\}} \quad (15)$$

such that (i) $A(Q_j)$ and $\tilde{A}(Q_j)$ are the exact answer and the approximate answer to Q_j respectively, and (ii) $sb(I_{DC})$ is the *sanity bound* computed on the input data cube I_{DC} , which is defined as follows:

$$sb(I_{DC}) = \frac{1}{100} \cdot \frac{\sum_{k=0}^{CN(I_{DC})} I_{DC}\{k\}}{CN(I_{DC})} \quad (16)$$

such that $CN(I_{DC})$ denotes the number of data cells contained in I_{DC} , and $I_{DC}\{k\}$ denotes the k -th data cell of I_{DC} .

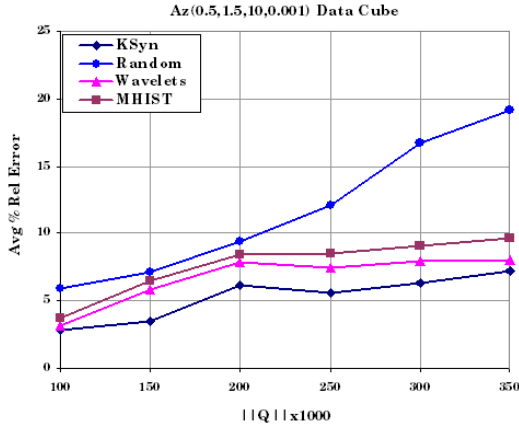


Figure 5. Variation of the average percentage relative error w.r.t. the query selectivity.

As comparison, we chose the following well-know-in-literature approximate query answering techniques: *MHist* histograms [24], wavelets [25], and sampling [14]. To compare the **KSyn** with the other techniques on a common basis, we impose the same space budget for all the methods for generating their own compressed representation of the target data cube. In our experiments, we set the space budget to be about the 8 – 9 % of the size of the data cube, as this order of magnitude for the compressed representation is widely accepted as a condition of “good performances”. Furthermore, for each comparison technique, we tried our best to set the configuration of the input parameters that the respective authors consider the best in their papers. This ensures a *fair* experimental analysis, i.e. an analysis such that each comparison technique provides its *best* performances, having fixed the space budget.

We engineered experiments aiming to test the **KSyn** performances by stressing it under the two specific limitations of approximate query answering in OLAP that have been recognized and discussed in Section 1: (i) scalability of the techniques w.r.t. increasing number of dimensions of the target data cubes, and (ii) need for guarantees on the degree of approximation of the answers. As regards the latter one, we highlight that, in Section 5, we demonstrated that **KSyn** allows us to provide probabilistically-bounded approximate answers. Thus, in the experimental phase, we tested the accuracy of the approximate answers in contrast to that provided by the comparison techniques. Other kinds of evaluation, mainly focused on the performances of the **KSyn**, such as build and access computational costs, query response time etc., are not handled in this paper, and must be considered as future work.

In the first experiment (see Figure 4), we measured the variation of \bar{e}_{rel} w.r.t. the number of dimensions of the data cubes defined through the $A_Z(0.5, 1.5, n, 0.001)$ instance by ranging n between $n_l = 5$ and $n_u = 30$. In the second experiment (see Figure 5), we measured the variation of \bar{e}_{rel} w.r.t. the selectivity of the $Q_S(P_j)$ query sets evaluated on the $A_Z(0.5, 1.5, 10, 0.001)$ data cube.

Experimental results shown in both Figure 4 and Figure 5 confirm the goodness of our proposal, even in comparison with the other similar techniques.

7. CONCLUSIONS AND FUTURE WORK

In this paper, important limitations of approximate query answering in OLAP have been recognized, and discussed in details, also starting from previous results provided by us in [9]. These limitations are: scalability of the techniques, and need for guarantees on the degree of approximation of the answers (i.e., the accuracy of the queries). Particularly, the paper focus on the latter limitation, whereas it refers to results shown in [9] for the former.

To face off limitations related to the accuracy of queries, a statistical framework for ensuring probabilistic bounds on the retrieved answers has been proposed, and tailored for the specific OLAP context, by taking advantage from the well-known Hoeffding’s inequality. Combining the results concerning dimensionality reduction of multidimensional data cubes, presented by us in [9], and this statistical framework leads to the definition of the **KSyn** synopsis data structure, which efficiently supports approximate query answering in OLAP, by also overcoming the recognized limitations.

Effective optimizations dealing with issues due to the lack of a-priori knowledge on both (i) the nature of the query-workloads against the target data cube and (ii) the kind of the data distribution characterizing the same data cube have also been designed and developed. Specifically, such optimizations aim to make the **KSyn** *query-conscious* [9]. Furthermore, an efficient algorithm for building and storing the **KSyn** has been proposed.

Preliminary experiments stating the goodness of the **KSyn**, also in comparison with other similar techniques, have been presented.

Future work is mainly focused on the following aspects: (i) comparing our technique against more recent proposals on wavelet-based compression of data cubes with probabilistic error guarantees [12,13]; (ii) performing further experiments for measuring query and maintenance performances of the **KSyn**; (iii) designing *hierarchical* MOLAP-based representation techniques for multidimensional data cubes, thus improving the query performances of the **KSyn**, and on issues concerning the maintenance of the **KSyn**, which is an exciting research challenge.

8. ACKNOWLEDGEMENTS

This work has been partially supported by the IST project (2001-33570) “*INFOMIX: Boosting Information Integration*”.

9. REFERENCES

- [1] S. Acharya, P.B. Gibbons, and V. Poosala, “AQUA: A Fast Decision Support System Using Approximate Query Answers”, *Proc. of VLDB*, pp. 754-757, 1999.
- [2] S. Acharya, V. Poosala, and S. Ramaswamy, “Selectivity Estimation in Spatial Databases”, *Proc. of ACM SIGMOD*, pp. 13-24, 1999.
- [3] B. Babcock, S. Chaudhuri, and G. Das, “Dynamic Sample Selection for Approximate Query Answers”, *Proc. of ACM SIGMOD*, pp. 539-550, 2003.

- [4] D. Barbarà, and X. Wu, “Loglinear-Based Quasi Cubes”, *Journal of Intelligent Information Systems*, Vol. 13, No. 3, pp. 255-276, 2001.
- [5] N. Bruno, S. Chaudhuri, and L. Gravano, “STHoles: A Multidimensional Workload-Aware Histogram”, *Proc. of ACM SIGMOD*, pp. 211-222, 2001.
- [6] F. Buccafurri, F. Furfaro, D. Saccà, and C. Sirangelo, “A Quad-Tree Based Multiresolution Approach for Two-Dimensional Summary Data”, *Proc. of IEEE SSDBM*, pp. 127-140, 2003.
- [7] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and R. Rastogi, “Overcoming Limitations of Sampling for Aggregation Queries”, *Proc. of IEEE ICDE*, pp. 534-542, 2001.
- [8] G. Colliat, “OLAP, Relational, and Multidimensional Database Systems”, *SIGMOD Record*, Vol. 25, No. 3, pp. 64-69, 1996.
- [9] A. Cuzzocrea, “Overcoming Limitations of Approximate Query Answering in OLAP”, *Proc. of IEEE IDEAS*, pp. 200-209, 2005.
- [10] A. Cuzzocrea, and U. Matrangolo, “Analytical Synopses for Approximate Query Answering in OLAP Environments”, *Proc. of DEXA*, pp. 359-370, 2004.
- [11] A. Cuzzocrea, W. Wang, and U. Matrangolo, “Answering Approximate Range Aggregate Queries on OLAP Data Cubes with Probabilistic Guarantees”, *Proc. of DaWaK*, pp. 97-107, 2004.
- [12] M.N. Garofalakis, and P.B. Gibbons, “Wavelet Synopses with Error Guarantees”, *Proc. of ACM SIGMOD*, pp. 476-487, 2002.
- [13] M.N. Garofalakis, and A. Kumar, “Deterministic Wavelet Thresholding for Maximum-Error Metrics”, *Proc. of ACM PODS*, pp. 166-176, 2004.
- [14] P.B. Gibbons, and Y. Matias, “New Sampling-Based Summary Statistics for Improving Approximate Query Answers”, *Proc. of ACM SIGMOD*, pp. 331-342, 1998.
- [15] D. Gunopulos, G. Kollios, V.J. Tsotras, and C. Domeniconi, “Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes”, *Proc. of ACM SIGMOD*, pp. 463-474, 2000.
- [16] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2000.
- [17] J.M. Hellerstein, P.J. Haas, and H.J. Wang, “Online Aggregation”, *Proc. of ACM SIGMOD*, pp. 171-182, 1997.
- [18] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant, “Range Queries in OLAP Data Cubes”, *Proc. of ACM SIGMOD*, pp. 73-88, 1997.
- [19] W. Hoeffding, “Probability Inequalities for Sums of Bounded Random Variables”, *Journal of the American Statistical Association*, Vol. 58, No. 301, pp. 13-30, 1963.
- [20] Y.E. Ioannidis, and V. Poosala, “Histogram-based Approximation of Set-Valued Query Answers”, *Proc. of VLDB*, pp. 174-185, 1999.
- [21] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.
- [22] M. Muralikrishna, and D.J. DeWitt, “Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries”, *Proc. of ACM SIGMOD*, pp. 28-36, 1988.
- [23] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer, “Towards an Analysis of Range Query Performance in Spatial Data Structures”, *Proc. of ACM PODS*, pp. 214-221, 1993.
- [24] V. Poosala, and Y.E. Ioannidis, “Selectivity Estimation without the Attribute Value Independence Assumption”, *Proc. of VLDB*, pp. 486-495, 1997.
- [25] J.S. Vitter, M. Wang, and B. Iyer, “Data Cube Approximation and Histograms via Wavelets”, *Proc. of ACM CIKM*, pp. 96-104, 1998.
- [26] J.S. Vitter, and M. Wang, “Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets”, *Proc. of ACM SIGMOD*, pp. 194-204, 1999.