Designing Random Sample Synopses with Outliers

Philipp Rösch, Rainer Gemulla, Wolfgang Lehner

Technische Universität Dresden
Faculty of Computer Science
Database Technology Group
01062 Dresden, Germany

{philipp.roesch, rainer.gemulla, wolfgang.lehner}@tu-dresden.de

November 2007
Technical Report

Designing Random Sample Synopses with Outliers

Random sampling is one of the most widely used means to build synopses of large datasets because samples retain the dataset's dimensionality. Unfortunately, the quality of the estimates derived from a sample is negatively affected by the presence of "outliers" in the data. In this paper, we show how to circumvent this shortcoming by constructing *outlier-aware sample synopses*. Our approach both improves and extends the well-known outlier indexing scheme: the former by reducing computational effort, the latter by supporting multiple aggregation columns. Since the estimation error typically differs for each aggregation column, we propose several different error measures that can be used to compare two or more synopses. These measures drive the design of our sampling algorithms: For a given amount of space and a given error measure, the goal is to find the synopsis with the lowest error. Due to the large number of possible synopses, our algorithms reduce the search space using heuristics; the synopsis is then constructed in a greedy way. As indicated by our experimental results on synthetic and real-world datasets, multi-column outlier indexing produces far more precise estimates than random sampling alone. Moreover, our algorithms scale well with respect to the size of the dataset and the number of aggregation columns.

1 Introduction

With its ability to provide lightweight summarizations of very large datasets, random sampling is ubiquitous in large-scale application scenarios. Its usage ranges from approximate query processing [1, 2, 5, 6, 9, 14] over query optimization [3, 4, 7, 8, 13] to data analysis [19, 20] and stream processing [17]. One of the main advantages of random sampling compared to other types of synopses is its ability to deal with a broad range of queries while providing probabilistic error bounds at the same time. In terms of relational operators, selection and aggregation (including grouping), for instance, can be applied on a sample in a straightforward manner. Consider the following query typical in OLAP environments:

```
SELECT YEAR, SUM(PRICE) AS REVENUE
FROM ORDERS
GROUP BY YEAR
```

This query computes the annual revenue from low-level transactional data, that is, the total value of each purchase order. In this paper, we develop algorithms to compute

synopses which are able to answer queries like the one above with accuracy and precision. The main challenge resides in the fact that estimates from random samples are sensitive to extreme values in the data. For the query above, there may be a small subset of transactions which contribute heavily to the annual revenue. For instance, imagine that the revenue in 2006 is $1 billion and that there is a single purchase order of $100 million. This order is represented as a single tuple in the database, but it constitutes 10% of the total revenue. We call tuples like this one *outliers*—they are significantly different from the rest of the data, but they are important for the result of the aggregation function. Thus, outliers should be well represented by a synopsis of the underlying dataset.

When random sampling is applied prior to aggregation, outliers in the data typically lead to underestimation (if they are not present in the sample) or overestimation (if present). Chaudhuri et al. proposed a sampling scheme called *outlier indexing* [5], which circumvents this effect by *storing outliers separately* and *sampling from the remaining part* of the dataset. Though outlier indexing significantly improves the precision of the sample synopsis, it is limited to a single predefined column. Obviously, a synopsis designed for, say, the precise estimation of the annual revenue (as above) may be inadequate for estimating other quantities like the amount of money lost by granting sales discounts. Another important challenge is to construct a synopsis which does not exceed a given amount of space. In this setting, one has to balance the space allocated for the sample and the space used for storing the outliers. To the best of our knowledge, there are no known solutions to this problem. In this paper, we answer the question:

> *Given a set of aggregation columns, can we efficiently compute an outlier-aware sample synopsis which requires no more than M tuples space?*

Here, the parameter M represents an upper bound on the sample size and influences both the synopsis' space consumption and the cost required to derive an estimate from it. Note that even though a tuple may have a "significant impact" on one of the aggregates, this does not necessarily hold for all the aggregates. Therefore, we have to be very careful in deciding which tuples are to be considered outliers and which are not. To address this problem, we propose several measures for quantifying the error of a synopsis. Since each of these measures has its own advantages and drawbacks, we compare them with each other. At the end of the paper, we recommend the measure with the best overall performance in our experiments.

After an error measure has been picked, the goal is to compute the optimum synopsis for the space constraint, that is, the synopsis which minimizes the error as defined by the measure. As shown later, finding the optimum synopses is computationally prohibitive, so that approximate solutions are needed. Our algorithm uses heuristics to prune the search space and selects outliers in a greedy manner. As indicated by our experiments, the estimation error induced by our synopses is close to the theoretical lower bound. Moreover, the estimation error can be further reduced by combining our synopsis with previous sampling techniques for group-by operations [1, 2].

The remainder of the paper is structured as follows: In Section 2, we give a brief overview of related work. In Section 3, we discuss what a "good synopsis" for aggregation queries should look like and introduce some measures to quantify these observations.

We present our algorithm for multi-column outlier indexing in Sections 4 and 5, and subsequently discuss further optimizations and maintenance techniques in Section 6. Experiments on synthetic and real-world datasets are presented in Section 7, while Section 8 concludes the paper with a summary of our results.

2 Related Work

Sampling has become an invaluable tool in application areas like approximate query processing [1, 2, 5, 6, 9, 14], query optimization [3, 4, 7, 8, 13], data analysis [19, 20] and stream processing [17]. Primarily, research focuses on methods to improve the quality of the estimates derived from the sample. Additionally, a vast amount of methods were proposed to keep the sample up-to-date when the underlying data evolves [10, 11, 12, 15, 22].

In the following, we give a brief overview on related work discussing the design of sample synopses trying to deal with data skew: Chaudhuri et al. proposed a technique called *outlier indexing* [5], which effectively reduces the negative impact of outliers in the data. The basic idea is to store a set of outliers accompanied by a random sample of the remaining part of the data. However, their approach applies to a single aggregation column only. Thus, we extend this technique by addressing the challenge of tuning a synopsis to an arbitrary number of different aggregates. Moreover, we introduce greedy algorithms which reduce the running time of the sampling algorithm by several orders of magnitude.

In [5, 6, 9], workload information is used to tailor the sample to the underlying application. However, workload-driven samples perform poorly on queries different from those in the workload and it is not known how database workloads (which consist of "exact" queries) can be leveraged to come up with queries relevant to approximate processing. In contrast, our algorithms do not require the presence of a workload itself; they operate on a high level description of the dataset to produce the synopses. It may be possible to combine workload-driven sampling schemes with our techniques: The weighted sampling scheme in [5] naturally extends to the algorithms presented in this paper.

Jermaine et al. [14, 16] propose a technique called *approximate pre-aggregation*, which leverages the availability of additional statistics on the data (for example, the revenue of a specific shop in a specific year). It is still open which statistics have to be gathered to provide good estimates, especially if the total space of the synopsis is restricted. Therefore, this technique is not applicable in scenarios where statistics have to be generated and stored together with the sample.

3 What is a Good Synopsis?

We start with a discussion of methods quantifying the error of a synopsis with respect to a pre-defined set of columns. Our methods allow for the comparison of different synopses and guide the algorithms presented in subsequent sections. In this paper, we limit our discussion to AVG and SUM aggregates. These aggregation functions are most

commonly used, especially in OLAP settings, and our techniques may apply to other aggregates as well.

Given a dataset $R = \{r_1, r_2, \ldots, r_{|R|}\}$ and a set of attributes $A_1, \ldots, A_l$, we quantify how well a synopsis Ψ of R can estimate aggregates on the A_j. The structure of our synopsis follows conventional outlier indexing [5]: Let $O \subset R$ be a set of outliers and S be a random sample from $R \setminus O$—the efficient choice of O is the main contribution of this paper. Then, $\Psi = (S, O)$ is an outlier-aware sample synopsis of the dataset R. The rationale behind this structure is that outliers are removed from the dataset prior to sampling, so their negative influence vanishes. To estimate an aggregate of the dataset using Ψ, we first estimate the aggregate for $R \setminus O$ using the sample S and second, we combine this estimate with the aggregate on the set O of outliers.

With M being the desired synopsis size, we require that $|\Psi| = |S| + |O| = M < |R|$, that is, the number of tuples in the synopsis is subject to a space constraint. The extraction of outliers effectively reduces the space available for the sample, so that we cannot extract an arbitrarily large amount of outliers. Thus, the main challenges are to decide on both the number and the composition of the outliers to extract and store in the set O.

3.1 Notation

This section briefly summarizes the notation used throughout this paper. We consistently use the index i for individual tuples of R and the index j for attributes of R. The quantity r_{ij} represents the value of attribute A_j of tuple r_i. Denote by $L_j(R) = \sum_{r_i \in R} r_{ij}$ the linear sum of the A_j's values. Then,

$$\mu_j(R) = \frac{L_j(R)}{|R|}$$

denotes the average value of the jth attribute in R. Similarly, the standard deviation of A_j can be expressed in terms of the quadratic sum $Q_j(R) = \sum_{r_i \in R} r_{ij}^2$:

$$\begin{aligned}\sigma_j(R) &= \sqrt{\frac{1}{|R|} \sum_{r_i \in R} (r_{ij} - \mu_j(R))^2} \\ &= \sqrt{\frac{Q_j(R)}{|R|} - \left(\frac{L_j(R)}{|R|}\right)^2}.\end{aligned}$$

This representation of $\mu_j(R)$ and $\sigma_j(R)$ has the advantage that both quantities can be incrementally maintained. We will leverage this fact in our algorithms later on. Next, let

$$RSD_j(R) = \frac{\sigma_j(R)}{|\mu_j(R)|}$$

denote the relative standard deviation of attribute A_j.[1] In contrast to the standard deviation, the RSD is unitless and can be compared across multiple columns.

[1]The relative standard deviation is not defined for $\mu_j = 0$ and may get very large for $\mu_j \approx 0$. In our implementation, we set $RSD_j(R) = \sigma_j(R)$ whenever $\mu_j \in [-1, 1]$.

With S being a uniform sample from R,

$$\hat{\mu}_j(R) = \frac{1}{|S|} \sum_{r_i \in S} r_{ij}$$

is an unbiased estimate of $\mu_j(R)$. Moreover, if $|S| = n$, the standard error (SE) of this estimate is:

$$\sigma_{\hat{\mu}_j}(R, n) = \sqrt{\frac{\sigma_j^2(R)}{n} \left(1 - \frac{n}{|R|}\right)} = \sigma_j(R) \sqrt{\frac{1}{n} - \frac{1}{|R|}}.$$

Since $\hat{\mu}_j(R)$ is unbiased, $\sigma_{\hat{\mu}_j}(R, n)$ equals the root-mean-square error (*RMSE*) of the estimate. Informally, the RMSE describes the deviation of the estimate from the exact value if sampling were to be performed multiple times. Clearly, a lower RMSE leads to more precise estimates, so that the error of a synopsis is directly tied to the standard error of the estimate.

As above, the relative standard error can be used to compare estimation errors across multiple columns. Since $\hat{\mu}_j(R)$ is unbiased, we can use $\mu_j(R)$ instead:

$$RSE_{\hat{\mu}_j}(R, n) = \frac{\sigma_{\hat{\mu}_j}(R, n)}{|\mu_j(R)|} = RSD_j(R) \sqrt{\frac{1}{n} - \frac{1}{|R|}}. \tag{1}$$

A value significantly smaller than 1 indicates low-error estimates.

Since all outliers are stored in their entirety, their average value can be determined without error. Thus the estimation error for aggregates over A_j is completely determined by the sample part of the synopsis:

$$RSE_{\hat{\mu}_j}(O) = RSE_{\hat{\mu}_j}(R \setminus O, |S|).$$

The value of $RSE_{\hat{\mu}_j}(O)$ is proportional to the relative standard deviation of the underlying dataset and inversely proportional to the square root of the size of the sample, see (1). This mirrors the challenge of deciding on the number of outliers to extract from the dataset prior to the sampling step: The more outliers we select, the smaller the standard deviation of the remaining data, but the smaller the sample size as well, which in turn increases the relative standard error of the estimate. Thus, the selection of an outlier has both a positive and a negative effect on the estimation error.

Example 1 *Consider the dataset shown in Table 1. It consists of 5 departments, the number of employees (E) and the number of projects (P) per department. The dataset contains two aggregation columns $A = \{E, P\}$, both having an average value of $\mu_E(R) = \mu_P(R) = 50$. We want to estimate the average of both columns using a synopsis Ψ, which contains exactly $M = 3$ tuples.*

The RSE of the estimate $\hat{\mu}_E$ is 15% *if O is empty,* 6% *if O contains DEP1, and* 4% *if O contains DEP1 and DEP5. Thus, if O is chosen adequately, outlier indexing is able to reduce the estimation error significantly. We refer to columns which benefit from outlier indexing as* skewed *columns. Though this is a very loose definition (e.g., it*

Table 1: Departments, number of employees and projects

DEPARTMENT	EMPLOYEES	PROJECTS
DEP1	10	10
DEP2	55	30
DEP3	60	50
DEP4	55	70
DEP5	70	90

depends on M), it is handy for the discussion of multi-column outlier indexing. Now, consider the RSE of $\hat{\mu}_P$: 21% if O is empty, 22% if O contains DEP1 or DEP5, and 27% if O contains both DEP1 and DEP5. Thus, it is not beneficial to extract outliers for aggregation on P; we refer to columns like this one as non-skewed *columns.*

In this example (and in most practical scenarios, of course), the optimum sets of outliers for the individual aggregates are different. In the following, *single-column outlier indexing* refers to the case of a single aggregate subject to optimization, while *multi-column outlier indexing* handles multiple aggregates simultaneously. After a brief review of the single-column case, we introduce measures which help in finding *one* synopsis optimized for *all* the columns in A, i.e., the multi-column case.

3.2 The Single-Column Case

In the single-column case, the synopsis Ψ is optimized for aggregations on a single attribute A of interest.[2] In this case, we can directly use $RSE_{\hat{\mu}_j}(O)$ as an error measure. Consequently, we design $\Psi = (S, O)$ so that $RSE_{\hat{\mu}}(O)$ is minimized. Chaudhuri et al. [5] have shown that O consists of the tuples on the lower and/or upper end of A's value range. A straightforward adaptation of their algorithm that simultaneously optimizes both the space allocation and the selection of outliers and that proceeds in a trial-and-error manner (referred to as *exhaustive approach*) requires $O(M^2 + |R|)$ time if R is sorted/indexed on A, or $O(M^2 + |R| \log M)$ time otherwise. We show how to decrease this cost in Section 4.

3.3 The Multi-Column Case

The situation gets more complex if multiple aggregates are subject to optimization. The problem is that the estimates of each aggregate have a different error, that is, the $RSE_{\hat{\mu}_j}$ are different ($j = 1 \ldots l$). If the synopsis is good for aggregates on a column A_1, it may be bad for another, say A_2. We introduce a set of measures which allow for the comparison of the error of two synopses. These measures depend on $RSE_{\hat{\mu}_1}(O), \ldots, RSE_{\hat{\mu}_l}(O)$, that is, the relative standard error of each estimate. We try to give insight on the advantages and disadvantages of each measure. Note that all measures are equivalent if applied to

[2] We use A and A_j synonymously in the single-column case.

the single-column case. Moreover, we ignore all columns which are constant, i.e., all columns A_j with $RSE_{\hat{\mu}_j}(\emptyset) = 0$.

Formally, a measure $\mathcal{M}$ is a function which takes a set of outliers as its input and which computes a numerical value for the error of the respective set of outliers, that is, $\mathcal{M} : \mathcal{P}_{M-1}(R) \to \mathbb{R}$ with $\mathcal{P}_k(R)$ denoting all subsets of R with at most k elements.[3] Note that the measure implicitly depends on both the relation R and the set of outliers O. A synopsis $\Psi = (S, O)$ is said to be *more precise with respect to* $\mathcal{M}$ than another synopsis $\Psi' = (S', O')$ if $\mathcal{M}(O) < \mathcal{M}(O')$, so that it is desirable to minimize the measure to achieve a low estimation error.

Perhaps the most intuitive measure is the maximum of the RSEs of the individual estimates:

$$\mathcal{M}_{MAX}(O) = \max_{j=1}^{l} RSE_{\hat{\mu}_j}(O).$$

This measure has the advantage that the column with the highest RSE is the main target of optimization. It has the disadvantage, however, that it only produces meaningful results if this column is skewed. To see this, suppose that the column with the largest RSE is non-skewed. Then, the value of $\mathcal{M}_{MAX}$ is smallest if no outlier is selected. The problem is that if there exist other columns which significantly benefit from outlier indexing, these columns do not influence $\mathcal{M}_{MAX}$ and remain undetected.

Instead of using the maximum, we may try to minimize the average of the relative standard errors of the estimates:

$$\mathcal{M}_{AVG}(O) = \frac{1}{l} \sum_{j=1}^{l} RSE_{\hat{\mu}_j}(O).$$

Again, this measure favors columns with a high RSD over columns with a low one (but not as heavily). To see this, suppose that we are able to reduce the RSD of a column A_1 by 1% by treating a tuple t_1 as an outlier, or alternatively, that the RSD of another column A_2 can be reduced by 10% by selecting t_2. If the RSD of A_1 is 10 times larger than the RSD of A_2, then t_1 is chosen, even though t_2 gives intuitively better results.

Going one step further, our final measure is independent from the absolute values of the relative standard errors of the estimates:

$$\mathcal{M}_{GEO}(O) = \sqrt[l]{\prod_{j=1}^{l} RSE_{\hat{\mu}_j}(O)} \propto \sqrt[l]{\prod_{j=1}^{l} \frac{RSE_{\hat{\mu}_j}(O)}{RSE_{\hat{\mu}_j}(\emptyset)}}.$$

Here, $RSE_{\hat{\mu}_j}(\emptyset)$ is the relative standard error achieved by simple random sampling and $\propto$ indicates proportionality. This measure is proportional to the geometric mean of the RSEs in each column relative to the RSE of simple random sampling, that is, it quantifies the improvement in the RSE compared to random sampling.[4] Intuitively, $\mathcal{M}_{GEO}$ directs lots of its optimization efforts to skewed columns, since estimates on these columns can

[3] We consider subsets with at most $M - 1$ elements, so that the sample is not empty.

[4] $\mathcal{M}_{GEO}(O)$ takes a value of 0 whenever one of the columns gets constant. Since this behavior is unwanted, an implementation might replace RSE by $\max(RSE, c)$, where c is a small constant.

be improved significantly. However, $\mathcal{M}_{GEO}$ may also put a lot of effort into columns which already have a very low RSE.

Example 2 *Recall the dataset shown in Table 1. If we optimize the synopsis for aggregates on both columns E and P and set $M = 3$ again, the optimum set of outliers for $\mathcal{M}_{MAX}$ is empty. If we use $\mathcal{M}_{AVG}$, DEP1 is chosen. Finally, $\mathcal{M}_{GEO}$ selects both DEP1 and DEP5.*

We experimented with a variety of other measures as well, but they performed poorly in practice. The measures defined above are the main device of multi-column outlier indexing. Using their ability to compare two synopses, we can develop algorithms which compute the best synopsis for the measure of choice. We defer further discussion of the advantages and drawbacks of each of the measures to Section 7, since they are somewhat data-dependent.

4 Algorithmic Description

We now describe the general algorithm for the computation of the set of outliers which minimizes a measure $\mathcal{M}$. Note that the algorithm presented below already decides on the composition and number of outliers; their actual computation is discussed in later sections. The algorithm is structured into 4 phases:

Phase 1: Initialization. In the first phase, we scan relation R once and build a random sample S_R of size M. If the cardinality of the relation is known a priori, we make use of sequential sampling [21]; otherwise, we use reservoir sampling [22]. Additionally, we construct a set $C \subseteq R$ of outlier candidates. All the tuples which are included in C are considered as potential outliers, so that this step effectively prunes the search space (if $|C| < |R|$). Finally, the quantities $L_j(R)$ and $Q_j(R)$ are computed, so that we can calculate means and standard deviations quickly and without accessing base data. Note that we do not access R again once the first phase has been completed.

Phase 2: Selection. In the second phase, we compute sets of outliers from C. Let $O_k \subseteq C$ be the optimum set of outliers (with respect to C) with $|O_k| = k$, that is, if exactly k outliers were chosen. The output of the second phase consists of the sets $O_0, \ldots, O_{M-1}$.

Phase 3: Decision. Decide on the optimum value k_{opt} for k. This is done by computing $\mathcal{M}(O_0), \ldots, \mathcal{M}(O_{M-1})$ and choosing k_{opt} so that $\mathcal{M}(O_{k_{opt}})$ is the minimum measure.

Phase 4: Finalization. In the last phase, we construct $\Psi = (S, O)$. We set $O = O_{k_{opt}}$. Now, we compute S from S_R by subsampling $S_R \setminus O$ down to size $M - k_{opt}$. Note that $|S| = M - k_{opt} \leq |S_R \setminus O|$. Thus, we do not have to access base data to compute S from S_R.

Clearly, the computation of the outlier candidates as well as the optimum set of outliers have the highest impact on the algorithm's performance. As a consequence, we restrict further attention to the first two phases of the algorithm. Note that the decision phase is inexpensive since

$$\sigma_j(R \setminus O_k) = \sqrt{\frac{Q_j(R) - Q_j(O_k)}{|R| - k} - \left(\frac{L_j(R) - L_j(O_k)}{|R| - k}\right)^2}.$$

In other words, $\sigma_j(R \setminus O_k)$ can be computed very quickly and without accessing R. We precompute this quantity without additional overhead while constructing the O_k in Phase 2.

The Single-Column Case

We now develop an instance of the above algorithm which optimizes the synopsis for a single aggregate. Recall that in the single-column case, the optimum set of outliers consists of tuples which lie on the lower and/or upper end of A's value range. Thus, in Phase 1, we restrict the candidate set C to the tuples with the $M-1$ smallest and $M-1$ largest values of A, thereby effectively reducing the search space from $|R|$ to $2(M-1)$ tuples. If R is sorted by A (or if there is an index), we are able to construct C in $O(M)$ time. Otherwise, we make use of two heaps of size $M-1$, thereby constructing C in $O(|R| \log M)$ time.

In Phase 2, a naive approach is to exhaustively try out all possible combinations (subject to the constraints above) to find the one which minimizes the estimation error. For each O_k, k possible allocations of candidates to actual outliers have to be investigated, which requires $O(M^2)$ time to determine $O_0, \ldots, O_{M-1}$. One might hope that there is a relationship between O_k and O_{k+1}, so that the O_k can be computed incrementally. However, the following negative result indicates that this is not the case:

Theorem 1 *In general, the relationship $O_{k-1} \subset O_k$ does not hold.*

Proof 1 *By counter-example. Consider the relation $R = \{1, 2, 3, 5, 1000, 1000, 1000\}$ and a synopsis size of $M = 6$. The optimum choices for the O_k are: $O_1 = \{1000\}$, $O_2 = \{1000, 1000\}$, $O_3 = \{1000, 1000, 1000\}$ but $O_4 = \{1, 2, 3, 5\}$. Clearly, $O_3 \not\subset O_4$.*

Most likely, someone looking at the above dataset will state that the 3 tuples with the value of 1000 are outliers, while the others are not. However, if we are allowed to select a very large number of outliers, it can be more effective to extract the "non-outlier fraction" of the data instead of the "real outliers." This procedure yields a more precise synopsis if the outliers have a low standard deviation among themselves. We argue that in any practical application, this case does not occur: On the one hand, the synopsis size is expected to be small with respect to R; on the other hand, outliers often do have a high standard deviation among themselves. In fact, in all our experiments on synthetic and real-world datasets, the relationship did hold, so that the above example is rather pathological. Therefore, we may use O_k to speed up the computation of O_{k+1}

in practice. This is the underlying idea of our greedy algorithm. For the sake of a clear notation, we denote approximations of O_k by $\hat{O}_k$.

The greedy algorithm starts with $\hat{O}_0 = \emptyset$. To compute $\hat{O}_k$ from $\hat{O}_{k-1}$, the tuple with the highest distance to the mean of the remaining dataset $R \setminus \hat{O}_{k-1}$ is added to $\hat{O}_{k-1}$. Clearly, this tuple is either the smallest or the largest one in $R \setminus \hat{O}_{k-1}$, and it is among the k largest and k smallest tuples of R. Since C is sorted, we can determine this tuple in $O(1)$ time. With k ranging from 0 to $M-1$, the greedy selection requires $O(M)$ time in total.

Example 3 *With $M = 3$, the outlier candidates of the employee column in Table 1 are set to the 2 smallest and 2 largest values: $C = \{10, 55, 60, 70\}$. Then, the exhaustive algorithm considers the following sets of outliers:*

O_0	$\underline{\emptyset}$
O_1	$\underline{\{10\}}$ and $\{70\}$
O_2	$\{10, 55\}$, $\underline{\{10, 70\}}$ and $\{60, 70\}$

The underlined sets are the choices of the greedy algorithm, which are optimal in this example. The exhaustive algorithm considers 6 possible outlier sets, while the greedy algorithm considers only 3.

To summarize, the exhaustive algorithm tries all possible combinations and requires $O(M^2 + |R| \log M)$ time to compute Ψ, while the greedy algorithm requires $O(M + |R| \log M)$ time. In most cases, the two algorithms produce exactly the same result. Note that all estimates derived from Ψ are unbiased and have relative standard error $RSE_{\hat{\mu}}(O)$ computed by (1), *even if we do not select the optimum set of outliers*. To that extent, it does not matter whether O is optimal or an almost optimum approximation.

5 Multi-Column Outlier Indexing

A straightforward extension of outlier indexing to multiple aggregation columns is to compute and store the outliers of all the columns separately. However, such an approach ignores the correlation between the columns and therefore may not produce optimal synopses. We follow a different strategy which extracts tuples with a significant impact on most (or all) of the aggregates. Before we discuss the details of our algorithm, we note that—in the multi-column case—we cannot get better results than in the single-column case.

Theorem 2 *The optimum set of outliers $O_{k_{opt}}$ in the single-column case provides a lower bound on the relative standard error of the estimate achievable by multi-column outlier indexing.*

This theorem directly follows from the optimality of $O_{k_{opt}}$.

To derive a multi-column outlier index, the naive approach is to set $C = R$, that is, to treat all tuples as outlier candidates. Then, to compute O_k, we may try all $\binom{|R|}{k}$ possible

sets of outliers and choose the one which minimizes our measure. Clearly, this exhaustive approach is not scalable. For example, to compute $O_{5,000}$ with $|R| = 1,000,000$, there are more than $8 \cdot 10^{13668}$ such sets. We therefore introduce (1) a heuristic strategy for the computation of the outlier candidates and (2) a greedy strategy (similar to the one above) for computing low-error approximations of the O_k. Note that all estimates derived from the resulting synopsis are unbiased.

5.1 Pruning the Search Space

In our approach, the search space for the optimum set of outliers is pruned during the computation of the candidates in Phase 1. The fewer candidates we select, the less space and time is required in the second phase of the algorithm. Though C should be small, care must be taken that C contains as many of the "real" outliers as possible—otherwise, the decision phase has inadequate input. The problem with computing the candidates is that the single-column algorithm presented above does not naturally extend to the multi-column case. If we simply include the $M-1$ smallest and largest values of each column into C, there is no guarantee that the optimum set of outliers can be computed. The reason lies in the correlation between both columns, as underlined in the following example.

Example 4 *Consider a relation R with two columns A_1 and A_2 consisting of the tuples $(4,20)$, $10x(5,5)$, $(19,19)$ and $(20,4)$, and set $M = 2$. For both columns A_1 and A_2, the tuples with the smallest and largest value are $C = \{(4,20),(20,4)\}$. However, the optimum choice for O_1 is the tuple $(19,19)$, which is not in C.*

Aside from non-optimality, this choice of C is prohibitive in terms of memory consumption: C can grow as large as $2l(M-1)$ tuples, which is infeasible for the large amount of columns found in practical scenarios. We propose a heuristic approach which selects exactly $M-1$ tuples from the base relation, *no matter how many columns are subject to optimization*. In general, these tuples satisfy one or both of the following properties: (1) the tuple has extreme values in *some* of the columns or (2) the tuple has values which differ from the average—though not necessarily extremely—in *most* of the columns. Our approach is to assign a *weight* to each tuple $r_i \in R$. The weight is chosen in such a way that it is large whenever (1) or (2) holds and small otherwise. Consequently, we include the tuples with the *largest* weights in the set C of outlier candidates.

The assignment of good weights to the individual tuples is crucial for our algorithm. Clearly, the weights should correlate with the measure we try to minimize. Inspired by the computation of the RSD, we might pick the tuples with the largest squared distances to the mean:

$$\mathcal{W}_{DistMean}(r_i) = \sum_{j=1}^{l} \left(\frac{r_{ij} - \mu_j(R)}{\mu_j(R)} \right)^2 .$$

If $\mathcal{W}_{DistMean}(r_i)$ is large for some tuple r_i, the tuple almost certainly satisfies (1) or (2).[5] The next two weights are related to the measures presented in Section 3:

$$\mathcal{W}_{SumRSD}(r_i) = -\sum_{j=1}^{l} RSD_j(R \setminus \{r_i\}),$$

$$\mathcal{W}_{ProdRSD}(r_i) = -\prod_{j=1}^{l} RSD_j(R \setminus \{r_i\}).$$

The weight of tuple r_i denotes the sum (product) of the RSDs for each column if r_i were removed from R. The minus sign ensures that smaller deviations produce larger weights. In fact, the weights are chosen such that for $\mathcal{M}_{AVG}$ ($\mathcal{M}_{GEO}$), the tuple with the largest value of $\mathcal{W}_{SumRSD}$ ($\mathcal{W}_{ProdRSD}$) is the optimum choice for O_1.

The time complexity for weight computation depends on the number of aggregates only, that is, it is independent of the database size and the synopsis size since:

$$RSD_j(R \setminus \{r_i\}) = \frac{\sqrt{\frac{Q_j(R)-r_{ij}^2}{|R|-1} - \left(\frac{L_j(R)-r_{ij}}{|R|-1}\right)^2}}{|\mu_j(R)|}.$$

In general, we estimate the quantities $L_j(R)$ and $Q_j(R)$ using column statistics. If there are no statistics available, we perform an additional table scan of R.

To extract the tuples with the largest weights, we make use of a heap data structure which is organized in ascending order of the tuple weights. At any time, the heap contains the tuples with the largest weights seen so far—the tuple with the smallest of these weights resides in the root node. Our algorithm therefore runs in $O(|R| \log M)$ worst-case time and requires $O(M)$ space.

Example 5 *Suppose that we want to compute the outlier candidates for aggregates on E and P in Table 1. Using $\mathcal{W}_{SumRSD}$, the weights for the individual tuples are as follows:*

r_i	DEP1	DEP2	DEP3	DEP4	DEP5
$RSD_E(R \setminus \{r_i\})$	0.12	0.46	0.45	0.46	0.41
$RSD_P(R \setminus \{r_i\})$	0.45	0.59	0.63	0.59	0.45
$\mathcal{W}_{SumRSD}(r_i)$	−0.57	−1.05	−1.08	−1.05	−0.86

According to the weights, we set $C = \{DEP1, DEP5\}$, in this order.

At first glance, one might expect that C is chosen perfectly, that is, that each of the optimal O_k is a subset of C. However, the weight computation is only optimal when no other outliers are selected. In practice, the benefit of selecting a tuple as an outlier depends on the other tuples in the set of outliers. For example, given that we want to

[5] We also experimented with squared distances from the median. Since median computation is expensive and results have been similar in our experiments, we do not present details about this weight.

select two outliers and that DEP1 has been selected already, DEP2 is a better choice than DEP4, even though both tuples have the same weight.[6] In any case, our experiments show that the heuristic approach produces synopses close to the lower bound, but it is several orders of magnitude faster than the exhaustive algorithm.

5.2 Outlier Selection

The set of outlier candidates contains only $M-1$ tuples using our heuristic approach. However, it is computationally prohibitive to try out all combinations for computing the O_k. In fact, there are $\binom{M-1}{k}$ possible allocations for each O_k, which sums to 2^{M-1} allocations in total, since k ranges from 0 to $M-1$. As in the single-column case, we propose a greedy algorithm which chooses one tuple at a time to advance from O_{k-1} to O_k. Again, we denote the so-defined outlier sets by $\hat{O}_0, \ldots, \hat{O}_{M-1}$ to underline the fact that a non-optimum set of outliers may be chosen. In our experimental study, we show that our greedy approach produces good results.

The outlier selection phase computes the O_k as follows: First, we set $\hat{O}_0 = \emptyset$. Then, we set $\hat{O}_k = \hat{O}_{k-1} \cup \{r\}$, where $r \in C \setminus \hat{O}_{k-1}$ is the tuple with the largest weight, so that the tuples are effectively selected in the order of their weights: $\hat{O}_k$ consists of the first k tuples in C. Though this procedure is fairly simple, it produces good results and is very efficient: Our algorithms require $O(M + |R| \log M)$ time and $O(M)$ space to construct a multi-column outlier index.

Example 6 *Recall that $C = \{DEP1,DEP5\}$ in the example scenario. Therefore, our algorithm sets $O_0 = \emptyset$, $O_1 = \{DEP1\}$ and $O_2 = \{DEP1,DEP5\}$. In Phase 3, the error measure for each of these outlier sets is computed: $\mathcal{M}_{AVG}(O_0) = 0.36$, $\mathcal{M}_{AVG}(O_1) = 0.29$ and $\mathcal{M}_{AVG}(O_2) = 0.31$. Thus, we set $k_{opt} = 1$ and $O = O_1$.*

6 Extensions

The sampling scheme presented in the previous sections optimizes a synopsis for a set of aggregates. So far, we assumed that aggregation is performed on the entire data set. However, in common OLAP applications, selection and grouping operations are conducted during aggregation. Though our synopses can be used to answer these queries, we are able to further reduce the estimation error by exploiting information on the intended use of the synopsis. Additionally, most data sets are not static, that is, they change over time. We outline techniques to maintain the synopsis by intercepting insertion and deletion requests to the underlying data set.

6.1 An Optimization for Group-by Queries

To compute the result of an aggregation query with group-by's from a random sample, one proceeds in a straightforward manner: The sample is partitioned into groups and the estimate is computed for each group independently. Though fairly simple, random

[6] DEP5 is the best choice, anyway.

sampling has the disadvantage of underrepresenting small groups, that is, groups which consist of only a few tuples. There are two known solutions to this problem: The first is based on stratification [1], while the second stores small groups separately [2]. In this section, we show how outlier indexing can be combined with the former approach, since the latter is not applicable for synopses with space constraints.

As indicated above, Acharya et al. [1] solved the small-group problem by partitioning (or stratifying) the data set prior to sampling. In more detail, given a set of grouping attributes, their *congressional sampling* algorithm creates a random sample of each occurring group; the available space is carefully distributed amongst the partitions. Multi-column outlier indexing can naturally be applied to the congressional sampling scheme by computing outliers per group instead of the whole dataset. The rationale behind this procedure is that outliers in the entire data set are not necessarily outliers in their respective group. Even if there are no small groups in the data, this combined approach produces far more precise synopses than outlier indexing or congressional sampling on its own (see Section 7).

6.2 Incremental Maintenance

If R is subject to insertions and deletions, Ψ can be maintained to reflect the changes on R. Our maintenance algorithm assumes that the optimum number of outliers (though not necessarily their composition) remains roughly constant. Recall that Ψ consists of two components: a uniform sample S and a set of outliers O. There exists a large body of work on maintaining uniform samples; e.g., the techniques in [11] allow us to maintain S if a tuple is added or removed from the non-outlier fraction of the dataset $(R \setminus O)$. We will not go into detail here but concentrate on the maintenance of the outlier set O. Whenever we say that a tuple is added to/removed from the sample, *we implicitly assume that a sample maintenance algorithm is used.*

Let k_{opt} be the number of outliers chosen during the computation of Ψ. For expository reasons, we show how to process deletions prior to discussing insertions. Suppose that a tuple r^- is deleted from the data set R. We remove r^- from the set of outliers if present, thereby reducing the current number k of outliers to a value less than k_{opt}. Otherwise, if r^- is not an outlier, we remove it from the sample.

Now, consider the insertion of a tuple r^+. If $k < k_{opt}$ due to deletions, we directly add r^+ to the set of outliers. Though it is likely that r^+ is not an outlier in our sense, adding r^+ to O always produces better results than adding it to the sample. Otherwise, whenever $k = k_{opt}$, we check whether the synopsis improves if the outlier with the highest score, say r^o, is replaced by r^+. If so, r^+ has a lower score than r^o: we remove r^o from O and add it to the sample instead. The old place of r^o is then assigned to r^+, which thereby becomes an outlier. Otherwise, if r^+ is not an outlier, there is no gain in precision if r^+ were included in O, so that we add it to the sample.

The above algorithm is able to detect new outliers, but it may have problems if a large number of outliers is removed from the data set. We argue that this situation occurs rarely in practice, since the outliers constitute a very small part of the underlying data set, so that most of the deletions will concern non-outliers. However, if O gets too small

due to deletions of outliers, one may recompute Ψ from scratch.

7 Experiments

We ran a variety of experiments in order to evaluate the efficiency of multi-column outlier indexing for aggregate estimation. We first evaluated the differences between the exhaustive and the greedy approach in terms of computational cost and precision of the estimates. Afterwards, we experimented with well-defined synthetic datasets, thereby discovering the influence of certain "data formations" on the error of the synopsis. Finally, we ran several experiments on a large real-world dataset consisting of retail data so as to validate the effectiveness of multi-column outlier indexing (MCOI) on real data with and without grouping. Our experiments indicate that MCOI performs significantly better than simple random sampling (SRS), while providing the same flexibility for query evaluation.

7.1 Experimental Setup

We implemented MCOI within our Derby/S prototype [18], which extends the Apache Derby database system with approximate query processing techniques. Most of the experiments were conducted on an Athlon AMD XP 3000+ system running Linux with 2GB of main memory. As a basis for our experiments on synthetic data, we generated three single-column datasets with $1,000,000$ values each, which are distributed as shown in Figure 1:

- D_N is a non-skewed dataset which is almost uniformly distributed and has been generated by subsequently sampling values from a standard normal distribution and accepting only values in $[-\frac{1}{2}, \frac{1}{2}]$.
- D_R has an asymmetric distribution and outliers on the right-hand side. This dataset has been constructed using the Student's t-distribution with one degree of freedom. Only positive values have been accepted.
- Finally, D_B is distributed symmetrically and has outliers on both ends. Again, it has been generated using a Student's t-distribution but this time, none of the values has been discarded.

With the exception of D_N, our datasets are heavily skewed, so that outlier indexing is thought to perform well. The main reason for using these datasets is to examine the relative performance of the individual measures. Note that we can artificially increase/decrease the relative standard deviation of each dataset by adding a constant value to all tuples.

Our real-world dataset consists of market research data and is made up of $4,383,694$ tuples and 5 columns. Each row represents an item available in a specific year (`YEAR`) together with its price (`PRICE`), the amount of items currently stocked (`STOCK`) and the number of sales (`SALES`) and purchases (`PURCHASE`) of the item.

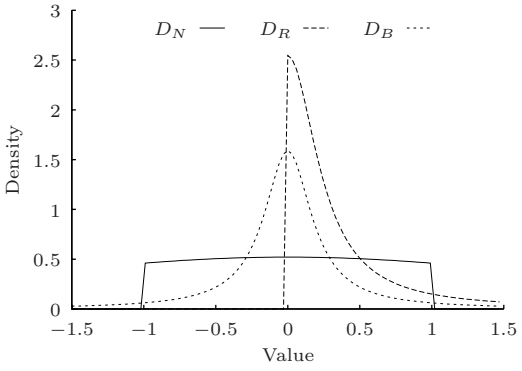


Figure 1: Density of synthetic data

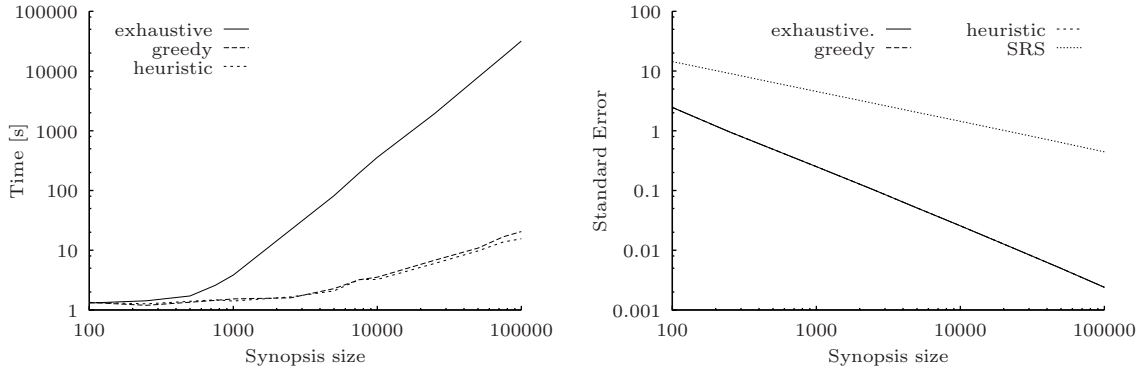
Unless stated otherwise, we use the average of the columns as aggregate function. Each experiment has been run several times and results have been averaged. Note that some of the figures make use of a logarithmic scale.

7.2 Exact vs. Approximate Computation

We compared the execution time of the sample computation as well as the error of the final synopsis for the exact and the approximate algorithms.

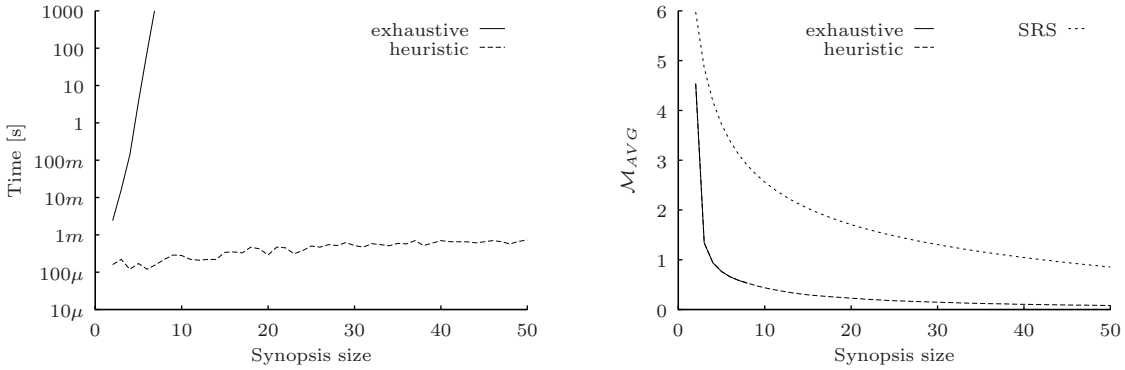
The single-column case. There are three different alternatives: a) the exhaustive approach—a straightforward adaptation of [5], b) the greedy algorithm presented in Sec. 4, and c) the heuristic MCOI algorithm of Sec. 5. Fig. 2a shows the execution time for each of the algorithms when applied to the D_B dataset. Note that the execution time is independent of the composition of the dataset, that is, running the algorithms on D_N or D_R produces exactly the same result. With increasing synopsis size, the approximate approaches—which run in $O(M + |R| \log M)$—were several orders of magnitude faster than the exhaustive algorithm—which runs in $O(M^2 + |R| \log M)$. All three approaches produced similar results on all three synthetic datasets; an example plot for D_B is shown in Fig. 2b. The exhaustive approach and the greedy approach always produced exactly the same results. This underlines the rather theoretic nature of the negative result in Theorem 1. The heuristic approach produced slightly less precise estimates, but the deviation from the optimum is too small to be visible in the plot. However, it required only half the memory and it worked with multiple aggregates.

The multi-column case. There are two different alternatives: a) the exhaustive approach and b) our heuristic MCOI algorithm. We were not able to compare these two algorithms on our synthetic datasets, since the exhaustive approach is far too slow to produce results in reasonable time. Instead, we used a sample of 100 tuples from D_B and D_R, joined both datasets randomly and computed a synopsis with the $\mathcal{M}_{AVG}$ measure on both columns. As shown in Fig. 3a, the exhaustive approach is very expensive: For $M = 8$, it considers $\sum_{i=0}^{7} \binom{100}{i} > 17$ billion possible outlier sets, resulting in an execution time of more than 6 hours. In contrast, our approximate solution required less than $1ms$ and scaled well with the size of the synopsis. Fig. 3b plots the $\mathcal{M}_{AVG}$ measure for



(a) Execution time (b) Final outcome

Figure 2: Results for the single-column case



(a) Execution time (multi-column case) (b) Final outcome (multi-column case)

Figure 3: Results for the multi-column case

the resulting synopses. Again, the difference of the errors is too small to be visible in the plot. Furthermore, the heuristic approach produced far more precise synopses than random sampling.

7.3 Comparing the Measures

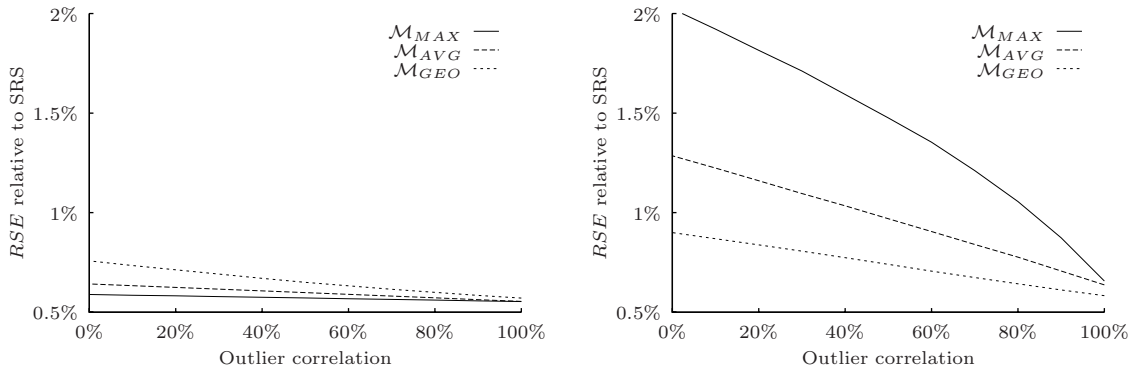
The focus of the following experiments is less to show the effectiveness of outlier indexing than to discuss the advantages and disadvantages of each measure.

Weights and measures. We first evaluated the impact of the weight computation on the error of the synopsis. Therefore, we joined the datasets D_B and D_R randomly and computed synopses for all combinations of measures and weights. Table 2 shows the results for a synopsis consisting of $100,000$ tuples. Each row has to be read separately: the best weight for each measure is marked with "Best"; numbers indicate relative deviations from the best case. As might be expected, measures and weights are closely related. The best combination of measures and weights, as given in the table, has been consistently observed in our experiments.

Multiple skewed columns. In this experiment, we evaluated the performance of

Table 2: Error of the measures with the different weights

	$\mathcal{W}_{DistMean}$	$\mathcal{W}_{SumRSD}$	$\mathcal{W}_{ProdRSD}$
$\mathcal{M}_{MAX}$	Best	+5.82%	+16.66%
$\mathcal{M}_{AVG}$	+3.55%	Best	+5.16%
$\mathcal{M}_{GEO}$	+78.80%	+20.69%	Best



(a) Col_B (higher RSD) (b) Col_R (lower RSD)

Figure 4: Multi-column outlier indexing on skewed data

MCOI on 2 skewed columns. We first scaled D_R and D_B to the relative standard deviations of $RSD_R = 10,000\%$ and $RSD_B = 100,000\%$. Then, we joined both sets in such a way that outliers in one dataset join with outliers in the other. We refer to the resulting columns as Col_R and Col_B. Since both columns are highly correlated with respect to their outliers, we randomly rearranged a varying fraction of the values in Col_B. By using this approach, we are able to quantify the outlier correlation between both columns: 100% correlation means that none of the values has been rearranged, while 0% correlation indicates that all the values have been shuffled, so that Col_R and Col_B are independent.

We computed a synopsis of 100,000 tuples for the resulting datasets. We ran the experiment with the $\mathcal{M}_{MAX}$, $\mathcal{M}_{AVG}$ and $\mathcal{M}_{GEO}$ measures and with different correlations between outliers. Figures 4a and 4b plot the relative standard error of the estimates in comparison to SRS. First, $\mathcal{M}_{MAX}$ tries to optimize the column with the higher RSD, leading to the best performance on Col_B but in turn performing worst on Col_R. The results for $\mathcal{M}_{AVG}$ are similar, but more effort is put on Col_R. Finally, $\mathcal{M}_{GEO}$ decreases the error of the estimates of both columns similarly, i.e., without favoring any of them. The more outliers are correlated, the better the estimate achieved by MCOI gets—no matter which measure has been chosen. However, even if outliers are uncorrelated, MCOI reduces the error significantly. In fact, it is less than 2.1% of the error of SRS in any case.

Synopsis size. Next, we evaluated the impact of the synopsis size on the relative standard error. We used the same setting as described above and fixed the correlation ratio to 60%. Fig. 5 compares SRS and MCOI with the $\mathcal{M}_{AVG}$ measure for different

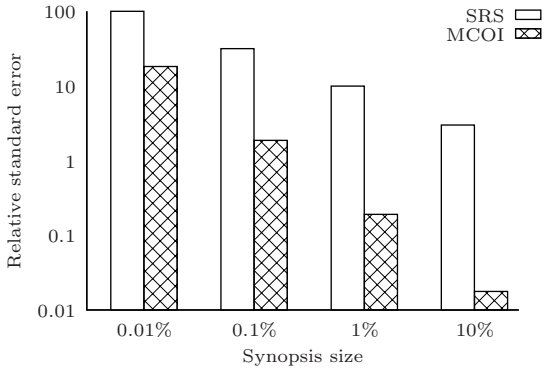


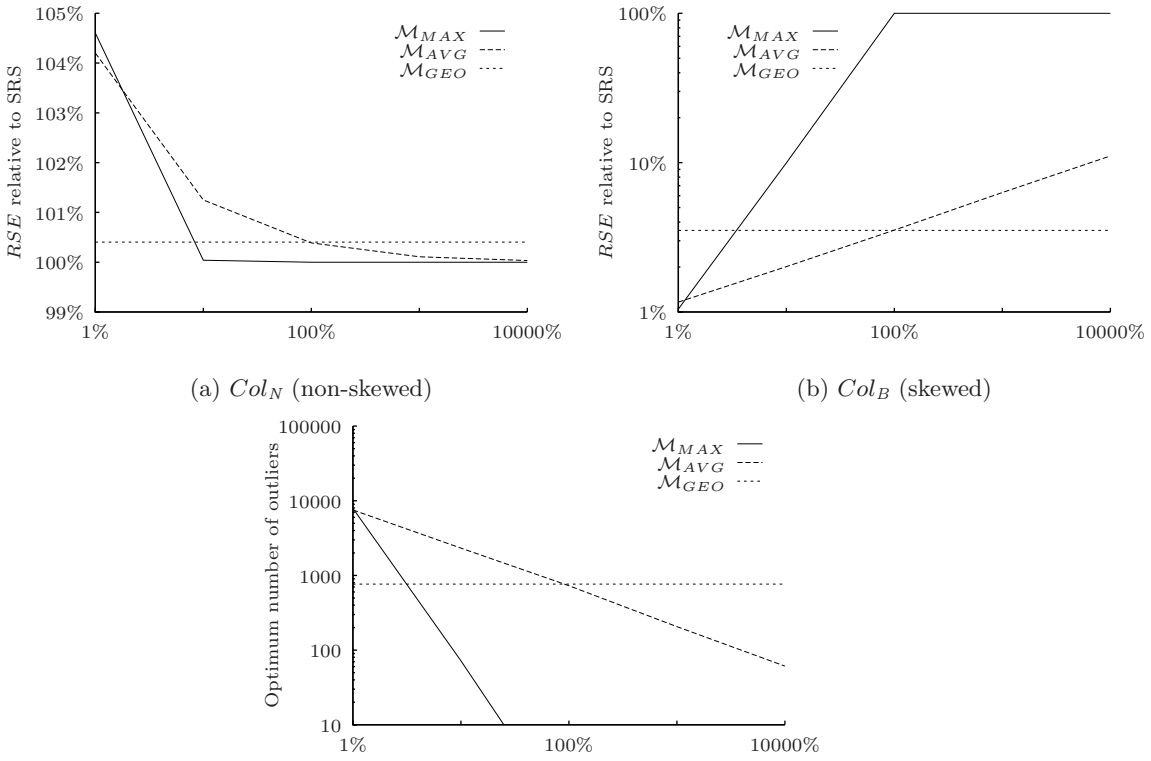
Figure 5: Influence of synopsis size

sizes of the synopsis (the figures for the other measures look similar). As can be seen, the effectiveness of outlier indexing increases with increasing synopsis size. Recall that each outlier reduces the remaining space for the sample. If the synopsis is small, the decrease in sample size by storing an additional outlier weighs heavier than the reduction in the relative standard deviation, while the opposite holds for large synopses. For example, if the synopsis constitutes only 0.01% of the whole dataset, then MCOI reduces the error to 20% compared to SRS, while a synopsis of 10% reaches 0.6%.

Impact of non-skewed columns. Non-skewed columns have a significant impact on the synopsis produced by minimizing each of the measures. To examine this effect, we joined the datasets D_N and D_B randomly. Next, we successively modified the RSD of the non-skewed column (RSD_N) and observed the impact on the synopsis computed by MCOI. We varied RSD_N from 1% up to 100,000% compared to that of the skewed column. Note that non-skewed columns typically have smaller relative standard deviations than skewed ones, but that this is no necessity. Figures 6a (Col_N) and 6b (Col_B) show the RSEs of the estimates compared to SRS for different values of RSD_N. As can be seen, $\mathcal{M}_{GEO}$ produced the same synopsis in every setting since it is independent from the absolute values of the relative standard errors. The estimates of Col_N are neither improved—it contains no outliers—nor significantly deteriorated. For all other measures, one finds that the higher the RSD of the non-skewed column, the less optimization potential remains for the skewed one. When the RSD of Col_N is low, both $\mathcal{M}_{MAX}$ and $\mathcal{M}_{AVG}$ incur additional errors between 4% and 5% while at the same time significantly increasing the precision of the estimate on Col_B. As expected, $\mathcal{M}_{MAX}$ performs poorly if there is a non-skewed column with high RSD. In this case, the effectiveness of $\mathcal{M}_{AVG}$ degrades as well, though less.

Figure 6c plots the optimum number of outliers for each of the measures. Again, $\mathcal{M}_{GEO}$ is unaffected by the RSD of the non-skewed column. However, $\mathcal{M}_{MAX}$ and $\mathcal{M}_{AVG}$ select fewer outliers when the relative standard error of the non-skewed column is high, since these measures try to minimize the average/maximum estimation error. In fact, $\mathcal{M}_{MAX}$ selects no outliers at all as soon as the relative standard error of Col_N exceeds that of the skewed column ($\geq 100\%$).

Conclusion. The $\mathcal{M}_{AVG}$ measure showed the best overall performance in our exper-



(a) Col_N (non-skewed)

(b) Col_B (skewed)

(c) Number of outliers

Figure 6: Multi-column outlier indexing on skewed and non-skewed data

iments. The reason is that $\mathcal{M}_{MAX}$ only focuses on the column with the highest RSE; whereas $\mathcal{M}_{GEO}$ does not consider (the absolute value of) the RSE at all. The $\mathcal{M}_{AVG}$ measure lies in between these two extreme cases: it focuses on columns with a high RSD and it is able to cope with non-skewed and skewed columns.

7.4 Results on Real-World Data

We conducted a variety of experiments on the sales dataset described above. We evaluated the error of estimation for the average of the `PRICE`, `STOCK`, `SALES` and `PURCHASE` column.

Single-column index. In this experiment, we created a single-column outlier index (SCOI) on one of the 4 columns of the dataset and evaluated how it affects aggregates on the other columns. We used a synopsis size of 10% of the dataset. The results are presented in Fig. 7. The x-axis denotes the column used for outlier-indexing; the different boxes represent aggregates on each of the individual columns. As can be seen, a single-column index on one of the columns deteriorates the estimation error for most of the other columns. A noteworthy exception is `PURCHASE` and `SALES`, which seem to be highly correlated.

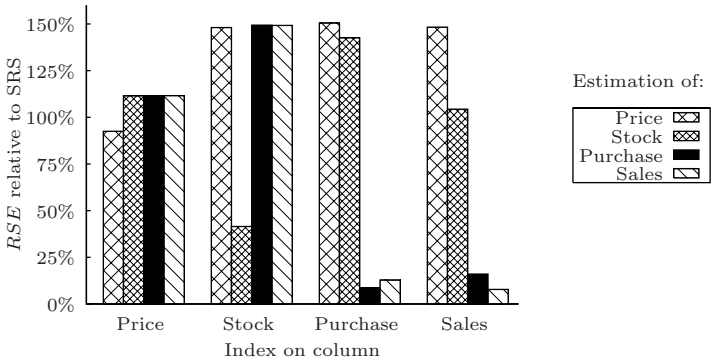


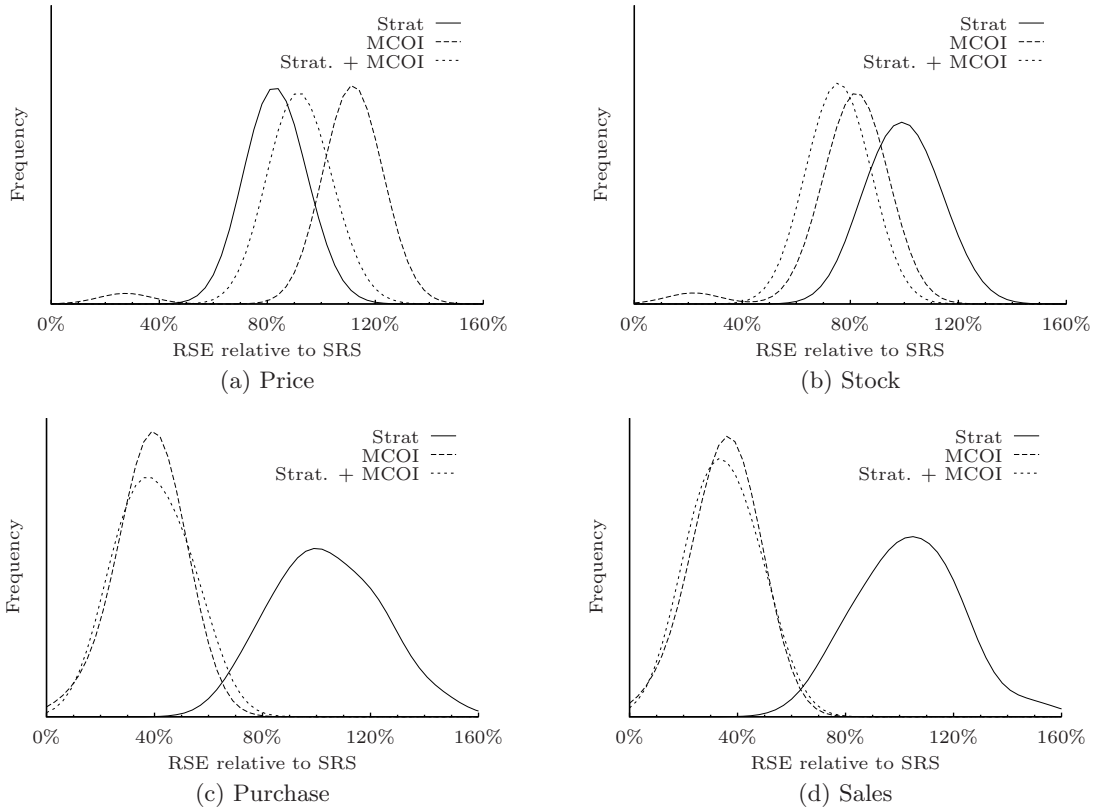
Figure 7: Single-column outlier indexing

Table 3: Relative standard error of the estimate

	Price	Stock	Purchase	Sales
SRS	0.133%	0.356%	1.577%	1.606%
$\mathcal{M}_{MAX}$	0.185%	0.210%	0.191%	0.147%
$\mathcal{M}_{AVG}$	0.169%	0.213%	0.191%	0.149%
$\mathcal{M}_{GEO}$	0.168%	0.214%	0.191%	0.149%
lower bound	0.123%	0.148%	0.136%	0.125%

Multi-column index. Next, we ran our MCOI scheme using the same synopsis size but this time optimizing all 4 columns together. As shown in Table 3, the different measures produced similar results. MCOI induced some additional estimation error on the non-skewed column (`PRICE`). However, since `PRICE`'s RSE is smaller by a factor of 2.6 to 12 compared to the other columns, this increase in RSE is more than outweighed by the decrease of the RSE for the other columns. We also gave the lower bound achieved by SCOI. As can be seen, the estimation error in the skewed columns gets close to this lower bound.

Grouping. In a final experiment we evaluated the impact of group-by operations on the error of the synopsis. We ran the experiment using SRS, a variant of the congressional sampling scheme [1] using proportional allocation (Strat.), MCOI, and a combined approach (Strat.+MCOI) applying the MCOI approach in each stratum. We computed the average value of the `PRICE`, `STOCK`, `PURCHASE` and `SALES` column for each of the 47 recorded `YEAR`s—group sizes varied from 68,088 to 120,566 tuples. Figures 8a to 8d show the distribution of the RSE relative to SRS for the different columns. First, stratification reduced the error in the `PRICE` column to about 80% of the error of SRS for most of the groups (Fig. 8a), since prices vary from year to year. Since `PRICE` is a non-skewed column, MCOI did not reduce the RSE for this column. In all the other columns, the estimate was not improved by stratification. However, MCOI significantly reduced the standard error to about 80% compared to SRS for the column `Stock` (Fig. 8b) and to about 35% compared to SRS for the columns `Purchase` (Fig. 8c) and `Sales` (Fig. 8d). Thus, MCOI produced good results if group-by queries were run on the synopsis, even



(a) Price (b) Stock

(c) Purchase (d) Sales

Figure 8: Multi-column outlier indexing w/ grouping

though it was optimized for aggregates on the entire dataset. The combination of stratification and MCOI further reduced the estimation error by combining the advantages of both approaches.

8 Summary

The presence of extreme values in the data has a negative impact on the error of estimates derived from random samples. We introduced several measures for the error of a synopsis, paying respect to the fact that synopses are often used to estimate a variety of aggregates. We extended the well-known outlier indexing scheme so that it produces synopses optimized for multiple aggregates simultaneously. We introduced a heuristic and greedy algorithm which handles the resulting combinatorial explosion of possible synopses and is by several orders of magnitude faster than the exhaustive approach. Our experiments showed that multi-column outlier indexing significantly reduces the estimation error caused by outliers in the data.

References

[1] S. Acharya, P.B. Gibbons, and V. Poosala. Congressional Samples for Approximate Answering of Group-By Queries. In *SIGMOD*, 2000.

[2] B. Babcock, S. Chaudhuri, and G. Das. Dynamic Sample Selection for Approximate Query Processing. In *SIGMOD*, 2003.

[3] P. Brown and P.J. Haas. BHUNT: Automatic Discovery of Fuzzy Algebraic Constraints in Relational Data. In *VLDB*, 2003.

[4] J.D. Brutlag and T. Richardson. A block sampling approach to distinct value estimation. Technical report, University of Washington, Department of Statistics, July 2000.

[5] S. Chaudhuri, G. Das, M. Datar, and R. Motwaniand V.R. Narasayya. Overcoming Limitations of Sampling for Aggregation Queries. In *ICDE*, 2001.

[6] S. Chaudhuri, G. Das, and V. Narasayya. A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries. In *SIGMOD*, 2001.

[7] S. Chaudhuri, G. Das, and U. Srivastava. Effective Use of Block-level Sampling in Statistics Estimation. In *SIGMOD*, 2004.

[8] D.J. DeWitt, J.F. Naughton, D.A. Schneider, and S. Seshadri. Practical Skew Handling in Parallel Joins. In *VLDB*, 1992.

[9] V. Ganti, M. Lee, and R. Ramakrishnan. ICICLES: Self-Tuning Samples for Approximate Query Answering. In *The VLDB Journal*, 2000.

[10] R. Gemulla and W. Lehner. Deferred Maintenance of Disk-Based Random Samples. In *EDBT*, 2006.

[11] R. Gemulla, W. Lehner, and P.J. Haas. A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Datasets. In *VLDB*, 2006.

[12] R. Gemulla, W. Lehner, and P.J. Haas. Maintaining Bernoulli Samples over Evolving Multisets. In *PODS*, 2007.

[13] I.F. Ilyas, V. Markl, P.J. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In *SIGMOD*, 2004.

[14] C. Jermaine. Robust Estimation With Sampling and Approximate Pre-Aggregation. In *VLDB*, 2003.

[15] C. Jermaine, A. Pol, and S. Arumugam. Online Maintenance of Very Large Random Samples. In *SIGMOD*, 2004.

[16] R. Jin, L. Glimcher, C. Jermaine, and G. Agrawal. New Sampling-Based Estimators for OLAP Queries. In *ICDE*, 2006.

[17] T. Johnson, S. Muthukrishnan, and I. Rozenbaum. Sampling Algorithms in a Stream Operator. In *SIGMOD*, 2005.

[18] A. Klein, R. Gemulla, P. Rösch, and W. Lehner. Derby/S: A DBMS for Sample-Based Query Answering (Demo). In *SIGMOD*, 2006.

[19] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. An Efficient Approximation Scheme for Data Mining Tasks. In *ICDE*, 2001.

[20] H. Toivonen. Sampling Large Databases for Association Rules. In *VLDB*, 1996.

[21] J.S. Vitter. Faster Methods for Random Sampling. *Commun. ACM*, 27(7), 1984.

[22] J.S. Vitter. Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software*, 11(1), March 1985.