A Framework for the Physical Design Problem for Data Synopses

Arnd Christian König and Gerhard Weikum

Department of Computer Science, University of the Saarland,
P.O. Box 151150, 66041 Saarbrücken, Germany
{koenig,weikum}@cs.uni-sb.de

Abstract. Maintaining statistics on multidimensional data distributions is crucial for predicting the run-time and result size of queries and data analysis tasks with acceptable accuracy. To this end a plethora of techniques have been proposed for maintaining a compact data "synopsis" on a single table, ranging from variants of histograms to methods based on wavelets and other transforms. However, the fundamental question of how to reconcile the synopses for large information sources with many tables has been largely unexplored. This paper develops a general framework for reconciling the synopses on many tables, which may come from different information sources. It shows how to compute the optimal combination of synopses for a given workload and a limited amount of available memory. The practicality of the approach and the accuracy of the proposed heuristics are demonstrated by experiments.

1 Introduction

Maintaining compact and accurate statistics on data distributions is of crucial importance for a number of tasks: (1) traditional query optimization that aims to find a good execution plan for a given query [4,17], (2) approximate query answering and initial data exploration [9,1,14,3,8], and (3) prediction of run-times and result sizes of complex data extraction and data analysis tasks on data mining platforms, where absolute predictions with decent accuracy are mandatory for prioritization and scheduling of long-running tasks. This broad importance of statistics management has led to a plethora of approximation techniques, for which [11] have coined the general term *"data synopses"*: advanced forms of histograms [24,12,16], spline synopses [18,19], sampling [5,13,10], and parametric curve-fitting techniques [27,7] all the way to highly sophisticated methods based on kernel estimators [2] or Wavelets and other transforms [22,21,3]. However, most of these techniques take the local viewpoint of optimizing the approximation error for a single data distribution such as one database table with preselected relevant attributes. The equally important problem which combination of synopses to maintain on the application's various datasets and how to divide the available memory between them has received only little attention [1,6,19], putting the burden of selecting and tuning appropriate synopses on the database administrator. This creates a *physical design problem for data synopses*, which

C.S. Jensen et al. (Eds.): EDBT 2002, LNCS 2287, pp. 627–645, 2002.

can be very difficult in advanced settings such as predicting run-times of data analysis tasks or information wealth of Web sources by a mediator. The state of the art is inadequate for a number of reasons:

- Since the accuracy of all approximation techniques depends on the memory size allotted to them, synopses for different data distributions compete for the available memory. In query optimization, for example, a small sized synopsis that does not improve query plan selection might have impact when given more memory.
- All proposed techniques are limited in the types of queries they support well. Most techniques aim at range-query selectivity estimation only and are thus unsuitable for complex queries with joins or aggregation/grouping unless additional synopses are maintained that are geared for approximations that cannot be inferred from the base representations (e.g., join synopses [1,19]). These additional synopses compete for the same memory space, and tuning the memory allocation for the various synopses is very difficult.
- Because the choice of an optimal combination of synopses is dependent on the workload (i.e., the query mix for which run-time and result size predictions or approximate answers need to be computed), it needs to be continuously adapted to the evolving workload properties.

1.1 Related Work

The reconciliation of different synopses as well as dedicated synopses for join queries (a uniform random sample over a foreign key join) was initially considered in [1]. Our work adapts these ideas and generalizes them, as their realization in the previous paper is limited to samples as base synopses and a data warehouse environment with a central fact table connected (via foreign keys) with the respective dimension tables. An extension of this approach to incorporate workload information (in the form of access locality) can be found in [8], but it is also limited to the above scenario.

The reconciliation problem for spline synopses was first discussed in [19], where a dynamic programming approach is proposed to minimize the error for a given set of synopses. However, this work offers no solution regarding which set of synopses to construct and does not take into account the characteristics of the workload. A similar approach for histograms was proposed in [15], extending [19] by offering heuristics that reduce the overhead of the dynamic programming problem. [6] considers a limited version of the problem: a set of synopses for query optimization are selected, based on whether or not they make a difference in plan selection. However, the approach is limited in a number of ways. Most importantly, synopsis selection is a series of yes-or-no decisions, with no consideration of the effect that variations of the size of a synopsis may have. This also has the consequence that the overall memory allotted to the selected synopses is utilized in a sub-optimal way. Furthermore, there is no consideration of special, dedicated (join-) synopses which do not constitute a (sub-)set of the attributes of a single relation.

1.2 Contribution and Outline

This paper develops a novel framework for the physical design problem for data synopses. Our framework covers the entire class of SPJ (i.e., select-project-join) queries. Note that projections are important also for predicting result sizes and run-times of grouping/aggregation, but have been almost completely ignored in prior work on data synopses. In contrast to the work in [6], our approach goes beyond the binary decisions on building vs. not building a certain synopsis, but also addresses the fundamentally important issue of how much memory each synopsis should be given. This is especially important when the role of statistics management goes beyond choosing good query execution plans, and synopses also serve to predict absolute run-times and result sizes, which in turn is highly relevant in data mining or Web source mediation environments. We characterize the exact solution for the optimal choice of synopses for a given workload. Taking into account the workload is a major step beyond our own prior work [19].

The remainder of this paper is organized as follows. In Section 2 we define the underlying optimization problem, briefly review the relevant parts from earlier work on spline synopses [19], and introduce our error model. Section 3 then describes how to determine the optimal set of synopses exactly, using two assumptions which (in earlier experiments) have been found to hold for nearly all datasets and which lead to a compact formulation of the necessary computations. In Section 4 we show how to combine the various building blocks of our framework into a unified algorithm. Section 5 contains an empirical validation of our approach in form of several experiments conducted with the *TPC-H* decision support benchmark. Finally, in Section 6 we summarize our work and give an outlook on future research.

2 Framework

We address the following optimization problem: given a number of datasets $\mathcal{R} := \{R_1, \ldots, R_n\}$ and a workload consisting of SPJ (select-project-join) queries $\mathcal{Q} := \{Q_1, \ldots, Q_k\}$, what is the best combination $\mathcal{S}$ of synopses such that the estimation error over all queries is minimized.

We assume that each query Q_i is mapped to exactly one synopsis $S_j \in \mathcal{S}$ which captures all attributes that are relevant for Q_i, i.e., attributes on which filter conditions are defined as well as attributes that appear in the query output. This is no limitation, as we can always decompose a more complex query into subqueries such that the above condition holds for each subquery. In fact, an SPJ query would often be the result of decomposing a complex SQL query (e.g., to produce an intermediate result for a group-by and aggregation decision-support query). The subqueries that matter in our context are those for which we wish to estimate the result or result size. In commercial query engines and in virtually all of the prior work on data synopses, these subqueries were limited to simple range selections. Our approach improves the state of the art in that we consider entire SPJ queries as the building blocks for data synopses.

So for a select-project query on a single dataset R we consider synopses that capture all of the selection's filter attributes and all attributes in the projection list (unless some of these attributes are considered irrelevant for the purpose of result approximation). For join queries that access attributes from multiple datasets $R_1, ..., R_l$ it is conceivable to construct a result approximation or result size estimation from multiple synopses. On the other hand, it is known that this approach may lead to unbounded approximation errors [5]. Therefore, we have adopted the approach of [1] to use special join synopses for this purpose. A join synopsis can be viewed as a regular synopsis that is derived from a virtual dataset that materializes the full result of the join. Such a materialized join view does not really have to be stored, but merely serves to construct the statistical data in the corresponding join synopsis.

2.1 Notation

We consider a set of relations $\mathcal{R} = \{R_1, \ldots, R_n\}$ and a set of queries $\mathcal{Q} := \{Q_1, \ldots, Q_k\}$. Each relation $R_i \in \mathcal{R}$ has at_i attributes $Att(R_i) = \{R_i.A_1, \ldots, R_i.A_{at_i}\}$. Queries can be "approximately answered" by a set of synopses $\mathcal{S} := \{S_1, \ldots, S_l\}$ corresponding to the data distributions $\mathcal{T}_1, \ldots, \mathcal{T}_l$; each S_i is the approximation of a relation R_p over the attributes $Att(S_i) \subseteq Att(R_p)$. Because in our context there is never more than one synopsis for a given set of attributes we also write S_x with x being the set of attributes captured by the synopsis, i.e., $S_{\{R_1.A_2, R_1.A_3\}}$ denotes the synopsis of R_1 over the two attributes A_2 and A_3. Analogously, we use the notation $\mathcal{T}_{\{R_1.A_2, R_1.A_3\}}$ to describe the corresponding joint data distribution in the full data set. The size of a synopsis S_x (in terms of the number of values necessary to store S_x) is denoted by $Size(S_x)$.

A simple (range) selection or projection query can be answered using the data distribution of the queried relation over the attributes involved in the range selection. A join query can be processed by examining the joint data distribution of the joining relations. Thus it is possible to assign to each query Q_i on relation R_p the *minimum* set $Min(Q_i)$ of attributes $\subseteq Att(R_p)$, whose corresponding data distributions must be examined to answer the query. For example consider the query q_1

$$\texttt{SELECT } R_1.A_1 \texttt{ WHERE } R_1.A_2 > 100.$$

This query can be answered by examining the joint data distribution of relation R_1 over the attributes $R_1.A_1$ and $R_1.A_2$, thus $Min(q_1) = \{R_1.A_1, R_1.A_2\}$.

When only the size of a result is of interest (for example in the context of query optimization), it is sufficient to query the attributes that determine the number of tuples in the result; assuming that no duplicate elimination is performed, in this case the minimum set becomes $Min(q_1) = \{R_1.A_2\}$. Consequently, the set $Min(Q_i)$ contains the information which synopses need to be built in order to answer query q_i while observing all correlations between the relevant attributes.

Concerning data distributions, we adopt the notation used in [24]. The *domain* $\mathcal{D}_{R_i.A_j}$ of a single attribute $R_i.A_j$ is the set of all possible values of $R_i.A_j$,

and the value set $\mathcal{V}_{R_i.A_j} \subseteq \mathcal{D}_{R_i.A_j}, \mathcal{V}_{R_i.A_j} = \{v_1, \ldots, v_n\}$, is the set of values for A_j actually present in the underlying relation R_i. The *density* of attribute X in a value range from a to b, $a, b \in \mathcal{D}_{R_i.A_j}$, is the number of unique values $v \in \mathcal{V}_{R_i.A_j}$ with $a \leq v < b$. The *frequency* f_i of v_i is the number of tuples in $\mathcal{R}$ with value v_i in attribute $R_i.A_j$. The *data distribution* of $R_i.A_j$ is the set of pairs $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}$. Similarly, a joint data distribution over d attributes $R_i.A_{j_1}, \ldots, R_i.A_{j_d}$ is a set of pairs $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}$, with $v_t \in \mathcal{V}_{R_i.A_{j_1}} \times \cdots \times \mathcal{V}_{R_i.A_{j_d}}$ and f_t being the number of tuples of value v_t.

2.2 Join Synopses

As pointed out in [5] (in the context of sampling), it is usually not feasible to estimate arbitrary join queries from approximations of the joining base relations with acceptable accuracy. For sampling, this phenomenon is discussed extensively in [5], but it does also hold for all other data reduction techniques that estimate join queries from approximations of the base relations.

For histograms, this is due to the fact that even small errors incurred when approximating the density of attribute values lead to drastic changes in the number and position of attribute values that find a join partner. This problem becomes worse in multi-dimensional histograms through the use of the assumption that, if $value_i$ unique attribute values are present in the i-th dimension within a bucket, then all $\prod_{l=1}^{\#dimensions} value_l$ combinations of these values are present [23]. Regarding join estimation via wavelets, consider the following example:

$$\mathcal{T}_1 = \{(v_1, 2), (v_2, 0), (v_3, 7), (v_4, 2)\} \quad \mathcal{T}_2 = \{(v_1, 10), (v_2, 10000), \ldots\}$$

Even if the approximation keeps all coefficients necessary to represent $\mathcal{T}_2$ and drops only a single coefficient of the representation of $\mathcal{T}_1$, the approximation of the join between the two distributions exhibits a large error, for the approximation $\hat{\mathcal{T}}_1 = \{(v_1, 1), (v_2, 1), (v_3, 7), (v_4, 2)\}$ now joins the 1000 $\mathcal{T}_2$ tuples with value v_2. The reason for this phenomenon is the fact that the *thresholding scheme* employed in [3] minimizes the overall mean squared error $\sum_{i=1}^{\mathcal{T}} (f_i - \hat{f}_i)^2$ for each relation, which minimizes the error regarding range selection queries, but disregards accurate join estimation.

As a solution, special synopses dedicated to estimating the data distribution resulting from a *foreign key* join were proposed in [1]. In [19] the issue was examined in the context of spline synopses and general equijoin queries; we proposed an algorithm that examines for each join the result of joining the base relations and adds special join synopses for join results. The trade-off to consider is that additional join synopses leave less memory for the synopses of the base relations.

Experiments showed that for virtually all examined datasets the addition of (even very small) join synopses improved the estimation quality greatly. Thus we adopt the following approach for join synopses: for all queries in $\mathcal{Q}$ involving joins, we add a "virtual relation" R' to $\mathcal{R}$ representing the joint data distribution

of the top node in the corresponding join tree (i.e. the complete n-way join if the join tree has n leaves). A query involving a join could thus be modeled by introducing join synopsis over the relevant attributes from the joining relations; consider query q_2:

$$\texttt{SELECT } R_1.A_1 \texttt{ FROM } R_1, R_2, R_3 \texttt{ WHERE } R_1.A_2 = R_2.A_3 \texttt{ AND } R_2.A_4 = R_3.A_5$$

Here we introduce $R' := R_1 \overset{A_2=A_3}{\bowtie} R_2 \overset{A_4=A_5}{\bowtie} R_3$. Then $Min(q_2) = \{R'.A_1\}$.

2.3 Spline Synopses

As the underlying statistics representation, we use *spline synopses*, which are described in detail in [18,19]. Our results also apply (with some adaptation) to other data reduction techniques (e.g. histograms). Spline synopses have particular properties that are advantageous in our physical design context. The approximation of a distribution $\mathcal{T}$ is again a data distribution $\hat{\mathcal{T}}$ with $|\mathcal{T}| = |\hat{\mathcal{T}}|$, i.e., for every attribute value pair (v_i, f_i) there is an approximate representation $(\hat{v}_i, \hat{f}_i)$. This makes it possible to use spline synopses for query estimation for virtually any query type, including more advanced operators such as top-k proximity search or spatial joins.

Spline synopses use two different approximation techniques for approximating attribute value density and attribute value frequencies. This means that the memory available for a single synopsis S over a distribution $\mathcal{T}$ is divided between the approximations based on each technique - the first one approximating the attribute frequencies, minimizing $\sum_{i=1}^{|\mathcal{T}|}(f_i - \hat{f}_i)^2$; the second technique approximates the value density, in case of a one-dimensional distribution minimizing $\sum_{i=1}^{|\mathcal{T}|}(v_i - \hat{v}_i)^2$. For d-dimensional distributions, a space-filling curve $\phi : [0,1]^d \mapsto [0,1]$ (more specifically, the Sierpiński curve) is employed to map each attribute-value $v_i \in \mathbb{R}^d$ to a value $v_i^l \in [0,1]$. We then approximate the latter values as $\hat{v}_{i,i=1\ldots n}^l$, minimizing $\sum_{i=1}^{|\mathcal{T}|}(v_i^l - \hat{v}_i^l)^2$. In order to use the resulting approximation for query estimation, the $\hat{v}_i^l$ are mapped back via ϕ^{-1} at query processing time. The key feature here is that the Sierpiński mapping preserves proximity, as it can be shown that

$$\forall v_i^l, \hat{v}_i^l \in [0,1] : \|\phi^{-1}(v_i^l) - \phi^{-1}(\hat{v}_i^l)\| \leq 2\sqrt{d+6}\ |v_i^l - \hat{v}_i^l|^{\frac{1}{d}} \text{ [26]}$$

with $\|\cdot\|$ denoting the L_2 norm (*Euclidian Distance*) between the data-points; i.e. by minimizing $|v_i^l - \hat{v}_i^l|$ we also reduce $\|\phi^{-1}(v_i^l) - \phi^{-1}(\hat{v}_i^l)\|$.

In this sense, the synopsis-construction process can be characterized as minimizing

$$\sum_{i=0}^{|\mathcal{T}|}(f_i - \hat{f}_i)^2 + r \cdot (\|v_i - \hat{v}_i\|)^2 \tag{1}$$

for an appropriately chosen r. Which values to choose for r is discussed in [19].

For both density and frequency, the resulting approximation is stored in buckets, with each bucket storing 3 values each: leftmost value, the start-point

and gradient of the frequency approximation (for frequency approximation), or leftmost value, number of distinct values and size of the interval between adjacent approximate values (for density approximation).

The above approach to capturing both value frequency and value density allows us to use a well defined error metric for the tuples contained in the result an arbitrary query (see the next section), whereas this is not easily possible for other techniques. For multi-dimensional distributions, histograms use the assumption, that all combinations of values in a bucket are realized [23], which generally leads to over-estimation of the number of distinct attribute values and under-estimation of their frequencies (see [19]). In wavelet-based approximation, the attribute-value distribution (i.e, density) is approximated only indirectly through the position of the values that have a frequency other than zero. Because the thresholding scheme used in [3] aims to minimize the overall mean square error $\sum_{i=0}^{|\mathcal{T}|}(f_i - \hat{f}_i)^2$ only, wavelet-based approximation generally does not result in a particularly accurate representation of the attribute-value density and thus cannot cope well with projection queries or grouping.

2.4 The Error Model

Our goal is to minimize the estimation error over all queries $Q_j \in \mathcal{Q}$. First consider a scenario in which all queries only depend on a single synopsis S_0 over the data distribution $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}$. We define the error for a given query $Q_j \in \mathcal{Q}$ by characterizing how well the query result $Result(Q_j) \subseteq \mathcal{T}$ is approximated. Then we define the error over all queries in $\mathcal{Q}$ with respect to a data distribution $\mathcal{T}$ as

$$Error(\mathcal{Q}, S_0) := \sum_{Q_j \in \mathcal{Q}} \Big(\sum_{i \in \{k | v_k \in Result(Q_j)\}} (f_i - \hat{f}_i)^2 + r \cdot (\|v_i - \hat{v}_i\|)^2 \Big) \qquad (2)$$

Thus, if we define $w_i := |\{Q' \mid v_i \in Result(Q'), Q' \in \mathcal{Q}\}|$, the sum of the errors for each query posed to synopsis S_0 can be written as:

$$Error(\mathcal{Q}, S_0) := \sum_{i=1}^{|\mathcal{T}|} w_i \cdot (f_i - \hat{f}_i)^2 + r \cdot w_i \cdot (\|v_i - \hat{v}_i\|)^2. \qquad (3)$$

Except for the weights w_i, this is the error function (equation 1) minimized by spline synopses. Since the weights w_i can be easily incorporated into the spline construction process, minimizing the query error in the case of a single distribution has become a problem of constructing the optimal spline synopsis, which has been solved in [19].

This is a slight simplification as it ignores approximation errors with regard to the boundary conditions of a query: when using a synopsis for answering a query some attribute values $\hat{v}_i$ may be included in the approximate answer even though the corresponding v_i would not be in the query result. Likewise, some attribute values may be erroneously excluded.

In a scenario with multiple synopses $\mathcal{S} := \{S_1, \ldots, S_l\}$, each query Q_j is answered (depending on $Min(Q_j)$) by a synopsis in $\mathcal{S}$. We use a mapping function $map : \bigcup 2^{\{Att(R)|R \in \mathcal{R}\}} \mapsto \{1, \ldots, l\}$ to assign each queried attribute combination to exactly one synopsis. We will describe how to obtain this mapping in Section 3.2. Note that this model assumes that queries over the same attribute combination are always mapped to the same synopsis (otherwise it would be necessary to store additional information on the mapping of specific queries, which would in turn compete for the memory available for synopses). Thus, the error over a set of synopses $\mathcal{S} := \{S_1, \ldots, S_l\}$ is defined as:

$$Error(\mathcal{Q}, \mathcal{S}) = \sum_{i=1}^{l} (Error(\{Q_j \in \mathcal{Q} \mid map(Min(Q_j)) = i\}, S_i).$$

Since the error of each synopsis S_i is dependent on the memory size $Size(S_i)$ of the synopsis, this is more accurately stated as:

$$Error(\mathcal{Q}, \mathcal{S})$$
$$= \min_{(Size(S_1),\ldots,Size(S_l))\in\mathbb{N}^l} \sum_{i=1}^{l} (Error(\{Q_j \in \mathcal{Q} \mid map(Min(Q_j)) = i\}, S_i)) \qquad (4)$$

under the constraint that $\sum_{i=1}^{l} Size(S_l)$ is equal to the memory size M available for all synopses together. Thus the problem of optimizing the estimation error for the entirety of queries in the workload can be seen as a problem of selecting the optimal set of synopses and choosing their sizes.

3 Synopsis Selection and Memory Allocation

To illustrate the issues involved in our method consider a workload $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, $\mathcal{Q}_1$ containing no_1 queries Q' (i.e., queries of type Q' whose fraction in the entire workload is proportional to no_1) with $Min(Q') = \{\{R_1.A_1\}\}$ and $\mathcal{Q}_2$ containing no_2 queries Q'' with $Min(Q'') = \{\{R_1.A_2\}\}$. Then these can be answered by either (a) two synopses $S_{\{R_1.A_1\}}$ and $S_{\{R_1.A_2\}}$ over each single attribute, or (b) one synopsis $S_{\{R_1.A_1,R_1.A_2\}}$ over the joint data distribution of $R_1.A_1$ and $R_1.A_2$. Therefore, to compute the optimal error for the overall available memory M we have to evaluate

$$Error(\mathcal{Q}) := \min \Big\{ \overbrace{Error(\mathcal{Q}, S_{\{R_1.A_1,R_1.A_2\}})}^{\text{Error for combination (b)}}$$
$$\underbrace{\min_{\substack{Size(S_{\{R_1.A_1\}})\in\mathbb{N} \\ Size(S_{\{R_1.A_2\}})\in\mathbb{N}}} \big(Error(\mathcal{Q}_1, S_{\{R_1.A_1\}}) + Error(\mathcal{Q}_2, S_{\{R_1.A_2\}})\big)}_{\text{Error for combination (a)}} \Big\}$$

(with $Size(S_{\{R_1.A_1,R_1.A_2\}}) = M$ and $Size(S_{\{R_1.A_1\}}) + Size(S_{\{R_1.A_2\}}) = M$) and keep track of the resulting synopses and memory partitioning. So we can

characterize the problem of computing the optimal set of synopses (and the corresponding memory allocation) as a two-step process:

(1) Computing $Error(\mathcal{Q}', S_x)$ for all candidate synopses S_x and all possible combinations of queries $\mathcal{Q}' \subseteq \mathcal{Q}$ that may be mapped to S_x and the maximum amount of Memory M to be used for S_x. This requires $O(M \cdot |\mathcal{T}_x|^2)$ steps (for optimal partitioning, see [19]) for each pair of S_x and $\mathcal{Q}'$ and also generates the values of $Error(\mathcal{Q}', S_x)$ for all values of $Size(S_x) \leq M$.

(2) Selecting the optimal set of synopses from a set of candidates computing an optimal memory partitioning such that the weighted sum over all synopsis errors (weighted by the number of times each synopsis is queried) becomes minimal for the synopses included in the optimal solution. Since the weights change according to the combinations of synopses in the solution, this is a different and more difficult problem than finding the optimal combination of synopses for different relations (which was solved by dynamic programming in [19]). As we will see in Section 3.2, the problem of synopsis selection and memory partitioning are closely related and thus solved together.

In the following (Sections 3.1 and 3.2), we will show how to solve the above problem for a single dataset $R \in \mathcal{R}$. The sub-solutions for all datasets in $\mathcal{R}$ can then be combined to solve the overall problem (Section 3.3).

3.1 Pruning the Search Space

Note that in the above example we never considered the combinations $\mathcal{S}' = \{S_{\{R_1.A_1\}}, S_{\{R_1.A_1,R_1.A_2\}}\}$ or $\mathcal{S}'' = \{S_{\{R_1.A_2\}}, S_{\{R_1.A_1,R_1.A_2\}}\}$. This is due to a simple property of spline synopses, which also normally holds for both histograms and Wavelet-based approximations:

Observation 1, "Pruning Property": When answering queries over the set of attributes a, a synopsis S_x, over the set of attributes x with $a \subseteq x$ will yield more accurate answers than a synopsis S_y if $x \subset y$ and both synopses are of identical size.

While artificial data distributions can be constructed that do not obey the above observation, we have found the pruning property to hold in all experiments on real-life datasets. The intuition behind it is the fact that by including more attributes in a synopsis, the number of unique attribute-value combinations v_i in the corresponding data distribution increases as well (in this respect, the synopsis selection problem is similar to the one of index selection), making it harder to capture all attribute values/frequencies with acceptable accuracy.

In the above case, it means that $S_{\{R_1.A_1\}}$ answers queries posed to $R_1.A_1$ better than $S_{\{R_1.A_1,R_1.A_2\}}$ (using the same memory). Similarly $S_{\{R_1.A_2\}}$ is an improvement over $S_{\{R_1.A_1,R_1.A_2\}}$ for queries posed to $R_1.A_2$. Thus the combination $\mathcal{S} = \{S_{\{R_1.A_1\}}, S_{\{R_1.A_2\}}\}$ generally outperforms $\mathcal{S}'$ or $\mathcal{S}''$.

Using the above observation, it becomes possible to characterize the set of candidate synopses in a compact manner. Consider a single relation R. Then the sets of attributes of R queried is $Syn(R, \mathcal{Q}) := \{Min(Q_i) \mid Q_i \in \mathcal{Q}\}$. Now the

set of all candidate synopses for R can be defined as:

$$Cand(R, \mathcal{Q}) := \{S_x \mid x = \bigcup z, z \subseteq Syn(R, \mathcal{Q})\}$$

The intuition behind the definition of $Cand(R, \mathcal{Q})$ is the following: if a Synopsis S_y is in $Cand(R, \mathcal{Q})$, it must be considered, for it is the most efficient way to answer a subset of queries of R using only one synopsis (all other synopses capable of answering the same subset would be less efficient, due to the pruning property). Conversely, if $S_y \notin Cand(R, \mathcal{Q})$ then y must be of the form $y = cand \cup nocand$ with $cand \in \{\bigcup z, z \subseteq Syn(R, \mathcal{Q})\}, nocand \subseteq Att(R), \forall x \in nocand : (cand \cup x) \notin \{\bigcup z, z \subseteq Syn(R, \mathcal{Q})\}$. But then S_{cand} answers the same set of queries as S_y and does so more efficiently, since $cand \subset y$. We further utilize a second observation for further pruning of the search space.

Observation 2, "Merge Property": For a set of queries $\mathcal{Q}$ each querying the same combination of attributes A, the error for answering the queries using one synopsis S over A with M memory is smaller than the error using two synopses S_1, S_2 over A, which together use memory M.

The intuition for this property is the following: By joining the synopses S_1 and S_2, the estimation for the (potentially) overlapping regions in S_1 and S_2 is improved, as additional memory is invested in its estimation. It is a trivial consequence that the merge property also holds for combinations of more than two synopses over A. In contrast to the pruning property, it is possible to prove that the merge property always holds (see [20] for the proof).

3.2 Selecting the Synopses for a Single Relation R

In the following we will describe, for a given set $\mathcal{Q}$ of queries over a single relation R, how to compute the optimal combination $\mathcal{S}$ of synopses, their sizes, and the corresponding mapping of queries, such that all queries can be answered and the overall error becomes minimal.

As shown before, the optimal combination of synopses $\mathcal{S}_{opt}$ can consist of synopses over single attribute combinations from $Syn(R, \mathcal{Q})$ that are optimal for a particular query in $\mathcal{Q}$, as well as synopses for the joint attribute combinations of multiple members of $Syn(R, \mathcal{Q})$, which are not optimal for any single query but more efficient than other combinations of synopses (using the same amount of memory) capable of answering the same queries. Now we want to capture this notion algorithmically, giving a method to construct a set of synopses for a given workload/data combination. We first introduce the necessary notation:

$Opt\_Syn_{\mathcal{A},M}$:= the combination of synopses for answering all queries over the attribute combinations in $\mathcal{A} \subseteq Syn(R, \mathcal{Q})$ using memory M as constructed below.

$Opt\_Err_{A,M}$:= the overall error resulting from $Opt\_Syn_{\mathcal{A},M}$.

Now consider the problem of computing the optimal combination of synopses $Opt\_Syn_{\mathcal{A},M}$ for given $\mathcal{A}$ and M. $Opt\_Syn_{\mathcal{A},M}$ has one of the following forms:

(a) $\mathbf{Opt\_Syn}_{\mathcal{A},\mathbf{M}} = \{\mathbf{S}_{\bigcup \mathcal{A}}\}$ with $Size(S_{\bigcup \mathcal{A}}) = M$ (one synopsis for all queries over the attribute combinations in $\mathcal{A}$).

(b) $\mathbf{Opt\_Syn}_{\mathcal{A},\mathbf{M}} = \mathbf{Opt\_Syn}_{\mathcal{A}',\mathbf{m}'} \cup \mathbf{Opt\_Syn}_{\mathcal{A}-\mathcal{A}',\mathbf{M}-\mathbf{m}'}$ (a combination of the optimal synopses for answering two disjoint subsets of $\mathcal{A}$ with $\mathcal{A}' \neq \emptyset$). Because of the merge property, we consider only decompositions for which $Opt\_Syn_{\mathcal{A}',m'} \cap Opt\_Syn_{\mathcal{A}-\mathcal{A}',M-m'} = \emptyset$.

Which combination is optimal depends on the error resulting from each alternative:

In case (a) $Opt\_Err_{\mathcal{A},M} = Error(\underbrace{\{Q' \mid Min(Q') = \bigcup \mathcal{A}\}}_{\text{The set of queries answered by } S_{\bigcup \mathcal{A}}}, \{S_{\bigcup \mathcal{A}}\})$

with $Size(S_{\bigcup \mathcal{A}}) = M$.

In case (b) $Opt\_Err_{\mathcal{A},M} = \min_{m' \in \{1,\ldots,M-1\}} Opt\_Err_{\mathcal{A}',m'} + Opt\_Err_{\mathcal{A}-\mathcal{A}',M-m'}$

Therefore, we can compute the optimal set of synopses for $\mathcal{A}$ by computing the minimal error for cases (a) and (b) and choosing the memory partitioning that minimizes the corresponding error. Note that by computing the optimal combination of synopses in the above manner, we implicitly also compute a mapping that dictates which attribute combinations from $Syn(R, \mathcal{Q})$ are mapped to which synopses: because of the above decomposition, $\mathcal{S} := Opt\_Err_{\bigcup Syn(R,\mathcal{Q}),M}$ is of the form $\mathcal{S} = \{S_{\bigcup \mathcal{A}_1}, \ldots, S_{\bigcup \mathcal{A}_l}\}$ with each $a \in Syn(R, \mathcal{Q})$ being a member of exactly one $\mathcal{A}_1, \ldots, \mathcal{A}_l$. While more complex models are possible in which queries over the same attribute combination are mapped to different members of $\mathcal{S}$, this would mean that additional information, from which the correct mapping for each single query could be derived at run-time, would have to be stored (creating contention for memory with the actual data synopses).

Using the above definitions, the final set of synopses kept for R using memory M is $Opt\_Syn_{\bigcup Syn(R,\mathcal{Q}),M}$, the corresponding error being $Opt\_Err_{\bigcup Syn(R,\mathcal{Q}),M}$. However, it is still necessary to prove that the optimal solution can indeed be obtained based on the decompositions described above:

Theorem: $Opt\_Syn_{\mathcal{A},M}$ constructed in the above manner is the *optimal* combination of synopses for answering all queries in $\mathcal{Q}$ over the attribute combinations in $\mathcal{A}$, when the *pruning* and *merge* properties hold.

Proof: We show that $\mathcal{S} := Opt\_Syn_{\mathcal{A},M}$ using the above construction implies that $\mathcal{S}$ is the optimal combination of synopses answering all queries over the attribute combinations in $\mathcal{A}$ using memory M. This is proven by induction over $|\mathcal{A}|$:

$|\mathcal{A}| = 1$: Then $Opt\_Syn_{\mathcal{A},M} = \{S_{\mathcal{A}}\}$ (no partitioning involving multiple synopses possible because of the merge property), and because of the pruning property $S_{\mathcal{A}}$ is the best way to answer queries over $\mathcal{A}$.

$|\mathcal{A}| \rightarrow |\mathcal{A}| + 1$: Now we assume that all $Opt\_Syn_{\mathcal{A},M}$ for $|\mathcal{A}| \leq h$ are indeed optimal and try to show the optimality for $Opt\_Syn_{\mathcal{A},M}$ with $|\mathcal{A}| = $ h+1. This is shown by contradiction:

Assumption: There exists a solution $\mathcal{S}_{opt} = \{S_{x_1}, \ldots, S_{x_t}\}$ (with $Size(S_{x_i}) = m_i, i = 1, \ldots, t$) such that the resulting overall error Err_{opt} over all queries is indeed smaller than $Opt\_Err_{\mathcal{A},M}$ with $\mathcal{S}_{opt} \neq Opt\_Syn_{\mathcal{A},M}$.
(Case 1) $|\mathcal{S}_{\mathbf{opt}}| = \mathbf{1}$**:** Then $\mathcal{S}_{opt} = \{S_{\mathcal{A}'}\}$, with $Size(S_{\mathcal{A}'}) = M$. Since $\mathcal{S}_{opt}$ has a smaller Error than $Opt\_Err_{\mathcal{A},M}$, $\mathcal{S}_{opt} \neq \{S_{\mathcal{A}}\}$ (as $S_{\mathcal{A}}$ is a possible synopsis combination for $Opt\_Syn_{\mathcal{A},M}$ and thus $Error(\mathcal{Q}, \{S_{\mathcal{A}}\}) \geq Opt\_Err_{\mathcal{A},M} > Err_{opt}$). However, since $\mathcal{S}_{opt}$ must be able to answer all queries over $\mathcal{A}$, $\mathcal{A} \subset \mathcal{A}'$ holds. Then it follows from the pruning property that $Opt\_Syn_{\mathcal{A},M}$ results in better accuracy than $\mathcal{S}_{opt}$, contradicting the previous assumption.
(Case 2) $|\mathcal{S}_{\mathbf{opt}}| > \mathbf{1}$**:** Because of the merge property, we assume that all queries to the same attribute combination $a \in \mathcal{A}$ are mapped to the same synopsis. Should this not be the case, we can replace $\mathcal{S}_{opt}$ by $\mathcal{S}'_{opt}$, for which all synopses over the same attribute have been merged, resulting in a smaller error. If we can now contradict the assumption for $\mathcal{S}'_{opt}$, we thereby contradict it for $\mathcal{S}_{opt}$, too.
Now $\mathcal{S}_{opt}$ can be written as $\mathcal{S}_{opt} = \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{S}_1 \neq \emptyset, \mathcal{S}_1 \neq \mathcal{S}, \mathcal{S}_2 := \mathcal{S} - \mathcal{S}_1$ with $\mathcal{S}_1 = \{S_{x_1^1}, \ldots, S_{x_p^1}\}, \mathcal{S}_2 = \{S_{x_1^2}, \ldots, S_{x_q^2}\}, p \leq h, q \leq h$. Because all queries over the same attribute combination are mapped to the same synopsis , both $\mathcal{S}_1$ and $\mathcal{S}_2$ each answer queries over the attribute combinations in disjoint subsets $\mathcal{A}_1, \mathcal{A}_2$ of $\mathcal{A}$ with $\mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}$. Then it follows from the induction hypothesis that $Opt\_Syn_{\mathcal{A}_1, \sum_{i=0}^{p} Size(S_{x_i^1})}$ results in a smaller error than $\mathcal{S}_1$ for queries over attribute combinations in $\mathcal{A}_1$, and $Opt\_Syn_{\mathcal{A}_1, \sum_{i=0}^{q} Size(S_{x_i^1})}$ results in a smaller error than $\mathcal{S}_2$ for queries over attribute combinations in $\mathcal{A}_2$. It follows that the error for $Opt_{\mathcal{A},M}$ is less than the one caused by $\mathcal{S}_{opt}$, contradicting the assumption. □

3.3 Selecting the Synopses for All Relations

The error over all relations for total memory size M can now be written as

$$Error(\mathcal{Q}, \mathcal{S}) = \min_{(M_1, \ldots, M_{|\mathcal{R}|}) \in \mathbb{N}^{|\mathcal{R}|}} \sum_{i=1}^{|\mathcal{R}|} Opt\_Err_{\bigcup Syn(R_i, \mathcal{Q}), M_i} \tag{5}$$

under the constraint that $\sum_{i=1}^{|\mathcal{R}|} M_i = M$. Note that this is equivalent to the initial definition in equation 4. Expression 5 can be solved by dynamic programming using $O(M^2 \cdot |\mathcal{R}|)$ operations. By keeping track of the memory partitioning $(M_1, \ldots, M_{|\mathcal{R}|})$, we can then determine the optimal set of synopses

$$\mathcal{S} := \bigcup_{i=1,\ldots,|\mathcal{R}|} Opt\_Syn_{\bigcup Syn(R_i, \mathcal{Q}), M_i}.$$

4 Putting the Pieces Together

Solving the *physical design problem* for data synopses can be characterized as a 4-step process:

1) Collection of workload information: We first need to acquire the necessary information about the access behavior of the workload, which can be done automatically by the data manager that processes the queries.
2) Enumeration of all possible synopsis combinations: As described in Section 3.3 the synopses selection problem can be solved for each relation independently; from the resulting sub-solutions the overall combination can then be obtained by solving equation 5. To obtain the sub-solution for each relation $R_i \in \mathcal{R}$, we first compute all possible synopsis combinations for $Opt\_Syn_{\mathcal{A},M}$ for R_i. This is done by traversing the lattice of the attribute combinations in $Cand(R_i, \mathcal{Q})$ in the order of the sizes $|\mathcal{T}_x|$ of the data distributions at each node. For each node we compute all synopsis combinations possible from its attribute combinations $\mathcal{A}_i$ and all subsets of $\mathcal{A}_i$ corresponding to nodes in the lattice (as well as potential mappings from queries to synopses).
3) Minimization of the error values: As described in Section 3.2, each of the combinations of synopses and mappings corresponds to an Opt_Err expression, which is defined in the form of a minimization problem. In order to determine the best synopsis combination, we have to compute the corresponding values for Opt_Err. This is done by constructing the corresponding synopses and evaluating the error for the resulting data distributions. The minimum Opt_Err expression corresponds to the optimal synopsis combination.

By combining the *enumeration* and *minimization* steps, it is furthermore possible to avoid solving identical minimization-problems more than once. Each time a new (sub-) combination of synopses/mapping is created the corresponding minimization problem is solved immediately. Because each new combination is either created by joining two previously know combinations together, plus at most one additional synopsis, the corresponding minimization problem can be solved using the solutions for the two joining synopses in at most $O(M)$ steps.
4) Construction of the final synopses: The overall optimal solution can now be obtained from the sub-solutions for each relation by minimizing equation 5.

The computational overhead of our techniques is caused by (a) the computation of the candidate synopses, (b) the solving of the resulting minimization problems, and (c) the enumeration of all possible minimization problems. A detailed discussion of the running times for (a) and (b) can be found in [19]. In order to assess the cost of (c), enumerating all minimization problems, we had our algorithm construct all possible synopses for a given set of attributes $\mathcal{A}$ for which all possible subsets were queried (i.e. $2^{|\mathcal{A}|}$ different types of queries and thus the same worst-case number of potential synopses). The running times for this worst-case stress test are shown in Table 1. Obviously, even though the space of all combinations grows exponentially with the size of $\mathcal{A}$, the enumeration is still reasonably efficient for up to 20 attributes, which covers the range of query-relevant attributes in most tables (including join views) in relational

Table 1. Running times for the enumeration on a *SUN UltraSPARC 4000* (168 Mhz)

# Attributes	Running time (sec.)	# Attributes	Running time (sec.)
4	0,009 sec.	12	1,23 sec.
8	0,049 sec.	16	93,93 sec.

databases. We have also developed a number of heuristic algorithms, that alleviate the potential bottlenecks arising from our approach. A detailed description of these can be found in the extended version of this paper [20].

5 Experiments

To validate our approach and to demonstrate both its accuracy and low overhead, we have implemented our techniques and applied them to a scenario based on the *TPC-H* decision support benchmark. We compared the synopses selection techniques introduced in this paper against several simpler heuristics. Because we are not aware of other approaches to the given problem, these heuristics are not intended to represent opponents. Rather, some represent assumptions commonly used in connection with synopses selection in commercial database systems. Others are used to examine how much approximation accuracy is affected by simpler approaches to either synopses selection or memory allocation.

5.1 Base Experiment

We used a subset of the queries of *TPC-H*, chosen to be large enough to make the synopses-selection problem non-trivial yet small enough to facilitate understanding of the resulting physical design. The queries selected were Q_1, Q_6, Q_{13}, Q_{15} and Q_{17}, referring to the LINEITEM, PART, ORDERS, and CUSTOMER tables[1]. Table 2 shows the query-relevant attribute sets, the *minimum sets* $Min(Q_i)$, for the above five queries. We chose these minimum sets according to a result-size approximation scenario, i.e., we only selected those attributes that are necessary to estimate the *number* of tuples in the query results (for queries which have an aggregation as the last operator, we estimate the result-size before the aggregation). This results in five multidimensional data distributions. Three of these are projections of the LINEITEM table, referred to as L onto subsets of its attributes (which all overlap so that there are a number of different, potentially suitable combinations of synopses). The other two data distributions to be approximated are join synopses J_1 = LINEITEM $\bowtie$ ORDERS and J_2 =LINEITEM $\bowtie$ PART. For our experiments, we used a scaled-down version of the *TPC-H* data with scale factor SF= $\frac{1}{100}$) and SF $*$ 500 KBytes memory available for all synopses together).

[1] The non-numerical values present in a TPC-H database are coded as numbers. For example, P.BRAND consists of a constant text string and two integers in the range $[1, 5]$. We only store the 25 possible number combinations.

Table 2. The *Mininmum Sets* for the used queries

Query	Min-Set
Q_1	L.SHIPDATE
Q_6	L.SHIPDATE, L.DISCOUNT, L.QUANTITY
Q_{13}	J_1.EXTENDED_PRICE, J_1.CLERK, J_1.DISCOUNT, J_1.RETURN_FLAG
Q_{15}	L.EXTENDED_PRICE, L.SHIPDATE, L.DISCOUNT
Q_{17}	J_2.CONTAINER, J_2.DISCOUNT, J_2.QUANTITY, J_2.BRAND

We compared the physical design technique presented in this paper against six heuristic competitors that were generated from the following two option sets for synopses selection and memory allocation.

Synopses selection:

Single. A single-dimensional synopsis was allocated for each attribute that appears at least once in the minimum sets of the five queries. While this heuristics cannot be expected to perform comparably to the more sophisticated allocation schema, we included it since most commercial database systems still use one-dimensional synopses/histograms only. So this heuristics gives an idea of the loss in accuracy when ignoring multi-attribute correlation.

Table. One multidimensional synopsis is allocated for each table, and this synopsis covers all attributes of the table that appear in the minimum sets. This heuristic results in a large single synopsis reflecting all correlations between attributes. However, because of the merge-property, the its accuracy may be significantly less than synopses using subsets of attributes.

and **Memory allocation:**

Uniform. Each synopsis is given the same size. Again, this assumption can be found in commercial database systems.

Tuples. The size of a synopsis is proportional to the size of the table that it refers to, measured in the number of tuples that reside in the table (where a join result is viewed as a table, too) multiplied with the number of attributes covered by the synopsis.

Values. The size of a synopsis is proportional to the size of the unique value combinations among the attributes over which the synopsis is built.

The synopsis-selection technique of this paper is referred to as *Opt_Syn*, the corresponding memory reconciliation as *Opt_Size*. To illustrate the importance of memory reconciliation for our overall approach, we also combined our synopsis-selection with the *Uniform*, *Tuples* and *Values*-based memory allocation; i.e., the optimal set of synopses was first generated and the sizes of these synopses were then computed using the above heuristics. For each set of synopses we executed 1000 instances of each query (using different, uniformly distributed, inputs for the query parameters, as specified in the benchmark) and used the available

synopses to estimate the result sizes. We measured the average relative error of the result sizes:

$$\textbf{Relative\_Error} := \frac{1}{n} \sum_{i=1,\ldots,n} \frac{|(exact\_size(i) - estimated\_size(i))|}{exact\_size(i)}$$

with n being the number of instances of all queries together. All queries occur with the same frequency in all experiments. The results of the first experiment are shown in the first three columns of Table 3.

Table 3. Error for the original *TPC-H*, skewed, and locally accessed data

Selection	Memory	Original data	Skewed data	Query locality
Single	Uniform	1.98	7.98	10.19
	Tuples	1.98	7.42	9.72
	Values	1.92	7.62	9.34
Table	Uniform	1.46	3.14	4.96
	Tuples	1.47	3.17	5.11
	Values	1.43	3.47	5.01
Opt_Syn	Uniform	1.05	1.14	1.04
Opt_Syn	Values	1.04	1.01	1.27
Opt_Syn	Tuples	1.03	1.08	1.17
Opt_Syn	*Opt_Size*	1.04	0.83	0.85

In this set of experiments, our technique employed for synopses selection had significant impact on the resulting approximation accuracy, whereas the way memory is allocated only results in negligible changes to the overall error.

5.2 Skewed and Correlated Data

As described earlier, for purposes of approximation it is crucial to preserve the correlation contained in the data. Unfortunately, the original TPC-H data is generated using uniformly random distributions for each attribute, resulting in almost completely uncorrelated data[2], which is not a good benchmark for data approximation techniques. Therefore, we ran a second set of experiments using the same schema, but with skewed and correlated data. This more realistic kind of data was generated the following way:

Skew in attribute-value frequencies. We generated the attribute-value frequencies so that the frequency of the attribute values was Zipf-like distributed;

[2] The exceptions being O.Totalprice (correlated with L.Tax, L.Discout, L.Extendedprice), L.Shipdate (correlated with O.Orderdate), L.Commitdate (correlated with O.Orderdate) and L.Receiptdate (correlated with L.Shipdate).

i.e., the frequency of the i-th most frequent value is proportional to $(1/i)^\theta$ where θ is a control parameter for the degree of skew. In this experiment we used $\theta = 1$.

Correlation between attributes. Here we permuted the generated data in order to obtain the desired correlation. After creating the data according to the TPC-H specification, we then performed (randomly chosen) permutations on the values of selected attributes in order to create specific correlations between pairs of attributes. The correlation itself is specified in terms of the linear correlation coefficient r_s [25]. For each of the following pairs of attributes we created data with a linear correlation coefficient $r_s \in [0.725, 0.775]$: (L.SHIPDATE, L.QUANTITY),(J_2.BRAND, J_2.CONTAINER), (P.PARTKEY, P.BRAND).

The results for this experiment are shown in the fourth column of Table 3. Again, the choice of the synopses-selection technique was most important with regards to the resulting approximation error: our *Opt_Syn* technique developed in this paper reduced the error by a factor of 7 and 3 compared to the Single and Table heuristics, respectively. In addition, with *Opt_Syn* for synopses selection, the use of our memory reconciliation technique *Opt_Size* resulted in noticeable further improvement. So for this more realistic dataset, the combination of *Opt_Syn* and *Opt_Size* outperformed all competitors by a significant margin.

5.3 Query Locality

We repeated the above experiments using a workload that exhibited significant locality, again using the data exhibiting significant skew and correlation. For this experiment, we generated the input parameters for the TPC-H queries using a Zipf-like distribution ($\theta = 0.25$), first executing 1000 queries of each type to obtain the weights w_i (see equation 3) we then used to construct the synopses. Subsequently, we ran another 1000 queries (with different parameters generated by the same probability distribution) for which we measured the error. The results for this experiment are shown in the fifth column of Table 3. The trends from the previous experiment can be observed here as well: the synopses-selection technique clearly outperforms the simpler approaches, with the estimation accuracy further improving when memory reconciliation is used.

6 Conclusions

In this paper we motivated and defined the *physical design problem for data synopses*. We proposed an algorithmic approach to its solution, discussed heuristics to alleviate computational bottlenecks, and provided an experimental evaluation. The experiments showed that the developed method achieves substantial gains over simpler heuristics in terms of the accuracy within the given memory constraint. They also showed that both aspects of our approach, synopses selection and tuning of the memory allocation, are important. Although we have carried out the derivation and implementation of our approach in the context of

spline synopsis, our approach is orthogonal to the specific form of synopses and applies equally well to histograms as well as other techniques (with some minor modifications).

References

1. S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join Synopses for Approximate Query Answering. In *Proceedings of the ACM SIGMOD Conference*, pages 275–286. ACM Press, 1999.
2. B. Blohsfeld, D. Korus, and B. Seeger. A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. In *Proceedings of the ACM SIGMOD Conference*, pages 239–250, 1999.
3. K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. In Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt, pages 111–122, 2000.
4. S. Chaudhuri. An overview of query optimization in relational systems. In *Proceedings of ACM PODS Conference*, pages 34–43, 1998.
5. S. Chaudhuri, R. Motwani, and V. R. Narasayya. On Random Sampling over Joins. In *Proceedings of the ACM SIGMOD Conference*, pages 263–274, 1999.
6. S. Chaudhuri and V. R. Narasayya. Automating Statistics management for Query Optimizers. *IEEE Conference on Data Engineering*, pages 339–348, 2000.
7. C. M. Chen and N. Roussoploulos. Adaptive Selectivity Estimation Using Query Feedback. In *Proceedings of the ACM SIGMOD Conference*, pages 161–172, 1994.
8. V. Ganti, M.-L. Lee, and R. Ramakrishnan. Icicles: Self-tuning samples for approximate query answering. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt*, pages 176–187, 2000.
9. P. B. Gibbons, S. Acharya, Y. Bartal, Y. Matias, S. Muthukrishnan, V. Poosala, S. Ramaswamy, and T. Suel. Aqua: System and techniques for approximate query answering. Technical report, Bell Labs, 1998.
10. P. B. Gibbons and Y. Matias. New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In *Proceedings of the ACM SIGMOD Conference*, 1998.
11. P. B. Gibbons and Y. Matias. Synopsis Data Structures for Massive Data Sets. In *Symposium on Discrete Algorithms*, 1999.
12. P. B. Gibbons, Y. Matias, and V. Poosala. Fast Incremental Maintenance of Approximate Histograms. In *Proceedings of the 23rd International Conference on Very Large Databases*, 1997.
13. P. J. Haas. Selectivity and Cost Estimation for Joins Based on Random Sampling. *Journal of Computer and System Sciences*, pages 550–569, 1996.
14. Y. E. Ioannidis and V. Poosala. Histogram-Based Approximation of Set-Valued Query-Answers. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 174–185, 1999.
15. H. Jagadish, H. Jin, B. C. Ooi, and K.-L. Tan. Global Optimization of Histograms. In *Proceedings of the ACM SIGMOD Conference*. ACM Press, 2001.
16. H. V. Jagadish, N. Koudas, S. Mutukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *Proceedings 24th International Conference on Very Large Databases*, pages 275–286, 1998.
17. N. Kabra and D. J. DeWitt. Efficient mid-query re-optimization of sub-optimal query execution plans. In *Proceedings of the ACM SIGMOD Conference*, 1998.

18. A. König and G. Weikum. Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-size Estimation. In *25th International Conference on Very Large Databases*, 1999.
19. A. König and G. Weikum. Auto-Tuned Spline Synopses for Database Statistics Management. 10th Int. Conference on the Management of Data, Pune, India, 2000.
20. A. König and G. Weikum. A Framework for the Physical Design Problem for Data Synopses (*extended version*) available at: `http://www-dbs.cs.uni-sb.de/`.
21. J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *Proceedings of the ACM SIGMOD Conference*, pages 205–214, 1999.
22. Y. Matias, J. S. Vitter, and M. Wang. Wavelet-Based Histograms for Selectivity Estimation. In *Proceedings of the ACM SIGMOD Conference*, pages 448–459, 1998.
23. V. Pooosala and Y. E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proceedings of the ACM SIGMOD Conference*, Athens, Greece, 1997.
24. V. Poosala. *Histogram-based Estimation Techniques in Database Systems*. PhD thesis, University of Wisconsin-Madison, 1997.
25. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Receipes in C*. Cambridge University Press, 1996.
26. E. Skubalska-Rafajlowicz. The Closed Curve Filling Multidimensional Cube, Technical Report no. 46/94. ICT Technical University of Wroclaw, 1994.
27. W. Sun, Y. Ling, N. Rishe, and Y. Deng. An instant and accurate Size Estimation Method for Joins and Selections in an Retrival-Intensive Environment. In *Proceedings of the ACM SIGMOD Conference*, pages 79–88, 1993.