

Approximate Query Processing with Confidence Bounds

October 25, 2013

Abstract

We present a new method to compute high quality approximations of aggregate database queries. Our method executes the queries on a fixed static sample of the original database, created off-line. It does not require any information about the underlying data distribution. The sample size does not depend on the size of the dataset but only on the complexity of the class of queries to be run, expressed through the VC-Dimension of the class. An upper bound to this quantity can be easily computed from the SQL expression of the queries. The values computed by running the queries on the sample are, with high probability, good approximations to the real values. To quantify the accuracy of the estimated value, we develop a method to compute confidence intervals around the estimation. This method is based on a convex optimization problem, which can be efficiently solved using known techniques. With high probability the confidence intervals contain the real values, for all queries. To our knowledge this is the first work that can achieve this guarantee for all queries in a given class simultaneously. We performed an extensive experimental evaluation of our methods, showing that they achieve high accuracy and great speed-up compared to running the queries on the original dataset.

1 Introduction

✓ **To-do:** Write.

Ideas:

- Aggregate queries gives you an intuition, a condensed piece of information, a rough picture of the data. The collection of statistics has somewhat an exploratory flavour.
- Computing them exactly is not worth it at takes a lot of time due to the desire to squeeze each possible amount of information from the DB.
- Given the goal, it is actually ok to settle for approximate answers. These still give enough information about the data, and they
- Online aggregation bad (slow)
- Guarantees are important. Not easy to achieve guarantees over all the queries. Need to use more powerful statistical techniques.

Our Contributions. In this work we introduce a new method for computing *high-quality approximate answers to aggregate queries*. We run queries on a *fixed static sample* of the dataset created off-line. The method does not make any assumption on the distribution of the data in the database, i.e., it is *distribution-free*. It does not requires any knowledge of the workload except for the *maximum complexity* of the queries that the user is interested in running. Our definition of complexity depends on the maximum number b of conditions in the selection predicate, the maximum number m of columns the maximum number of joins u . Starting from this information, our method derives a sample size such that a random sample of the database of that size can be used to compute good approximations of the values of all the queries with the specified complexity by running the queries on the sample. The sample fits in main memory and is computed only once off-line and only needs to be updated after major changes in the original database. This

results in a huge speedup in the execution of the queries. Given two user-specified parameters ε and δ which control the accuracy and the confidence of the approximate answers, the sample size is $O(\varepsilon^{-2}(\text{poly}(b, m, u) + \log(1/\delta)))$ and it does not depend on the size of the original database, but only on the maximum query complexity and on ε and δ . Deriving such a sample size is possible thanks to an application of VC-dimension theory to database queries. In order to be more informative about the quality of the approximate answers, we also developed a method based on compute *deterministic confidence intervals* for the answers. The confidence intervals can be computed efficiently, as the method requires the solution of a *convex optimization problem*. We can guarantee that, with probability at least $1 - \delta$, the computed interval will contain the true value of the aggregate query for all queries *simultaneously*. We conducted an extensive experimental evaluation of the accuracy of our estimations and confidence intervals and of the speedup achieved by our method, showing that it is very accurate and fast.

✓ **To-do:** We are missing stuff like “we are the first to do this and that”.

Outline. We introduced the basic definitions, concepts and tools in Sect. 2. Our method for computing good approximations of aggregated queries is presented in Sect. 3. We conducted an extensive experimental evaluation of our approach and report the results in Sect. 4. Sect. 5 contains a review of the relevant previous work. We wrap up with conclusions and future work in Sect. 6.

2 Preliminaries

In this Section, we introduce the necessary terminology, concepts, and tools that we will use to develop our results in the following sections.

2.1 Databases and Queries

We outline here some basic definitions about databases, queries, and selectivity. We refer the reader to complete textbooks for additional information [8]. We consider a *database* \mathcal{D} of k tables $\mathcal{T}_1, \dots, \mathcal{T}_k$. A *table* \mathcal{T} is a two-dimensional representation of data. It contains a number of rows or *tuples*, and may (and typically does) have multiple *columns*. We denote a column C of a table \mathcal{T} as $\mathcal{T}.C$ and, for a tuple $t \in \mathcal{T}$, the value of t in the column C as $t.C$. We denote the domain of the values that can appear in a column $\mathcal{T}.C$ as $D(\mathcal{T}.C)$. Each tuple t is then an element from the Cartesian product of the domains of the columns of \mathcal{T} , $t \in D(\mathcal{T}.C_1) \times \dots \times D(\mathcal{T}.C_m)$. Table 1 shows two examples of database tables, *Customers* and *CarColors*.

<i>Customers</i>					<i>CarColor</i>	
<i>Name</i>	<i>Age</i>	<i>Street</i>	<i>ZipCode</i>	<i>PhoneNumber</i>	<i>Name</i>	<i>Color</i>
John Doe	20	Pratt Av.	02906	401-1234567	John Doe	Blue
Jim Clark	30	Morris Rd.	02906	502-8902134	Jane Doe	Red
Greta Garbo	60	Pratt Av.	05902	853-9876543	Greta Garbo	Red

Table 1: Example of database tables.

Definition 1. Given a table \mathcal{T} with columns $\mathcal{T}.C_1, \dots, \mathcal{T}.C_\ell$, a *selection query* q on \mathcal{T} is an operation which returns a subset S of the tuples of \mathcal{T} such that a tuple t of \mathcal{T} belongs to S if and only if the values in t satisfy a condition \mathcal{C} (the *selection predicate*) expressed by q . In full generality, \mathcal{C} is the Boolean combination of clauses of the form “ $\mathcal{T}.C_i \text{ op } a_i$ ”, where $\mathcal{T}.C_i$ is a column of \mathcal{T} , “op” is one of $\{<, >, \geq, \leq, =, \neq\}$ and a_i is an element of the domain of $\mathcal{T}.C_i$.

As an example, the selection query

```
SELECT * FROM Customers WHERE Customers.ZipCode = 02906;
```

would return the first and second row of Table 1.

We assume that, for each column \mathcal{T}_C of each table \mathcal{T} , the domain $D(\mathcal{T}.C_i)$ is such that it is possible to build a total order relations on it. This assumptions does not exclude categorical domains from our discussion, because the only meaningful values for “op” for such domains are “=” and “ \neq ”, so we can just assume an arbitrarily but fixed order for the categories in the domain.

Definition 2. Given two tables \mathcal{T}_1 and \mathcal{T}_2 , a *join query* q on a common column C (i.e., a column present both in \mathcal{T}_1 and \mathcal{T}_2) is an operation which returns a subset of the Cartesian product of the tuples in \mathcal{T}_1 and \mathcal{T}_2 . The returned subset is defined as the set $\{(t_1, t_2) : t_1 \in \mathcal{T}_1, t_2 \in \mathcal{T}_2, \text{ s.t. } t_1.C \text{ op } t_2.C\}$ where “op” is one of $\{<, >, \geq, \leq, =, \neq\}$. The condition $\mathcal{T}_1.C \text{ op } \mathcal{T}_2.C$ is known as the *join predicate* \mathcal{C} .

An example of a join query is the following:

```
SELECT * FROM Customers, CarColors WHERE Customers.Name = CarColors.Name
```

This query would return the following tuples:

```
(John Doe, 21, Pratt Av., 02906, 401-1234567, Blue),
(Greta Garbo, 60, Pitman St., 05902, 853-9876543, Red)
```

The column *Name* is reported only once for clarity.

Our definition of a join query is basically equivalent to that of a *theta-join* [8, Sect.5.2.7], with the limitation that the join condition \mathcal{C} can only contain a single clause, i.e., a single condition on the relationship of the values in the shared column C and only involve the operators $\{<, >, \geq, \leq, =, \neq\}$ (with their meaning on $D(C)$). The pairs of tuples composing the output of the join in our definition have a one-to-one correspondence with the tuples in the output of the corresponding theta-join.

Definition 3. Given a set of ℓ tables $\mathcal{T}_1, \dots, \mathcal{T}_\ell$, a *combination of select and join operations* returns a subset of the Cartesian product of the tuples in the sets S_1, \dots, S_ℓ , where S_i is the output of a selection query on \mathcal{T}_i . The returned set is defined by the selection queries and by a set of join queries on S_1, \dots, S_ℓ . The condition \mathcal{C} is the *select/join predicate*, a Boolean combination of selection and join predicates.

As an example, the query

```
SELECT * FROM Customers, CarColors WHERE Customers.Name = CarColors.Name
AND Customers.ZipCode = 02906 AND CarColors.Color = Red
```

combines select and joins operations. It returns no tuple (empty answer), as there is no individual reported in both tables with zipcode 02906 and a red car.

When not further specified, we use the term “query” to denote a combination of select and join operations.

Definition 4. Let \mathcal{D} be a set of tables $\mathcal{D} = \{\mathcal{T}_1, \dots, \mathcal{T}_\ell\}$, and $\mathcal{G} = \{G_1, \dots, G_k\}$ be a subset of the columns in the tables of \mathcal{D} . For each G_i , let $V(G_i)$ be the set of values in the column G_i of the tuples in the tables G_i belongs to (it may be $V(G_i) \subset D(G_i)$). A *group-by query* q is a pair (q^*, \mathcal{G}) where q^* is a query on \mathcal{D} and \mathcal{G} is defined as above. The output of q is a *partition* \mathcal{P} of the output of q^* such that there is a class $S_v \in \mathcal{P}$ for each element v of the Cartesian product $V(\mathcal{G}) = V(G_1) \times V(G_2) \times \dots \times V(G_k)$. Each set S_v contains the tuples from the output of q^* with values corresponding to v on the columns G_1, \dots, G_k .

A group-by query (q^*, \mathcal{G}) can be expressed in SQL as

```
SELECT * FROM  $\mathcal{T}$  WHERE  $\mathcal{C}$  GROUP BY  $\mathcal{G}$ ;
```

where \mathcal{C} is the selection/join predicate of q^* .

For example, the group-by query

```
SELECT * FROM Customers WHERE Customers.Street = "Pratt Av." GROUP BY Customers.ZipCode
```

would return two sets, one for the value “02906” in *Customers.ZipCode*, containing the first tuple of the *Customers* table, and one for the value “05902” containing the third tuple.

Fact 1. Let $q = (q^*, \mathcal{G})$ be a group-by query as in Def. 4. We can see the output of q as the collection of the outputs of the queries in the set $\text{GQ}(q)$ where $q_g \in \text{GQ}(q)$ is a (non group-by) query such that $g = (g_1, \dots, g_k) \in \mathcal{V}(\mathcal{G})$ and the selection/join predicate of q_g is the selection/join predicate of q , denoted as \mathcal{C} , AND'ed with conditions involving the components of g :

$$\mathcal{C} \text{ AND } G_1 = g_1 \text{ AND } \dots \text{ AND } G_k = g_k. \quad (1)$$

Definition 5. Let $\mathcal{D} = \{\mathcal{T}_1, \dots, \mathcal{T}_\ell\}$ be a set of tables, and let $\mathcal{T} \in \mathcal{D}$, with a column $\mathcal{T}.C$. Let $M(\mathcal{T}.C)$ be the multiset of values taken by tuples in \mathcal{T} on the column $\mathcal{T}.C$. An aggregate query q is a pair (q^*, f) , where q^* is a (potentially group-by) query on \mathcal{D} , and f is a function that maps subsets of $M(\mathcal{T}.C)$ to members of \mathbb{R} . The output of q is the value taken by the function f when applied to the subset of $M(\mathcal{T}.C)$ in the tuples in the output of q^* . If q^* is a group-by query, f is computed for every group and the output of q is a set of values.

Typical examples for the function f include sum (SQL operator SUM), average (AVG), variance (VAR), standard deviation (STDEVP [13]). There's STDEV and STDEVP. It's the usual $1/n$ vs $1/(n-1)$. Similar for VAR and VARP. Should take care of this.], and median and quantile computation [14]. Are there standard SQL expressions for these?]. These are typically integrated in the query processing engine of commercial database management systems. An example of an aggregate query is the following:

```
SELECT AVG(Customers.Age) FROM Customers WHERE Customers.ZipCode = 02906;
```

The output of this query would be “25”, the average between the values in the *Age* column in the first two tuples of the *Customers* table.

Fact 2. Let f be a function as in Def. 5, and let A be a subset of $M(\mathcal{T}.C)$. We can see A as a collection of pairs (v, c) , where $v \in D(C)$ and c is a non-negative integer representing the number of times v appears in A . So, f can be seen as a function from $D(C) \times \mathbb{N}_0$ to \mathbb{R} .

✓ **To-do:** Add example with AVG.

A key concept that we will use throughout the paper is that of *selectivity* of a query.

Definition 6. Given a query q , the *cardinality* of its output is the number of elements (tuples if q is a selection queries, pairs of tuples if q is a join query, and ℓ -uples of tuples for combinations of join and select) in its output.

The *selectivity* $\sigma(q)$ of q is the ratio between its cardinality and the *product* of the sizes of its input tables.

2.2 VC-Dimension

The Vapnik-Chernovenkis (VC) Dimension of a space of points is a measure of the complexity or expressiveness of a family of indicator functions (or equivalently a family of subsets) defined on that space [23]. A finite bound on the VC-dimension of a structure implies a bound on the number of random samples required for approximately learning that structure. We outline here some basic definitions and results and refer the reader to the works of Alon and Spencer [2, Sect. 14.4], Mohri et al. [18] and Vapnik [22] for more details on VC-dimension. See Sect. 5 for applications of VC-dimension in computer science.

We define a *range space* as a pair (X, R) where X is a (finite or infinite) set and R is a (finite or infinite) family of subsets of X . The members of X are called *points* and those of R are called *ranges*. Given $A \subset X$, The *projection* of R on A is defined as $P_R(A) = \{r \cap A : r \in R\}$. If $P_R(A) = 2^A$, then A is said to be *shattered* by R . The VC-dimension of a range space is the cardinality of the largest set shattered by the space:

Definition 7. Let $S = (X, R)$ be a range space. The *Vapnik-Chervonenkis dimension* (or *VC-dimension*) of S , denoted as $\text{VC}(S)$ is the maximum cardinality of a shattered subset of X . If there are arbitrary large shattered subsets, then $\text{VC}(S) = \infty$.

Note that a range space (X, R) with an arbitrary large set of points X and an arbitrary large family of ranges R can have a bounded VC-dimension. A simple example is the family of intervals in $[0, 1]$ (i.e., X is all the points in $[0, 1]$ and

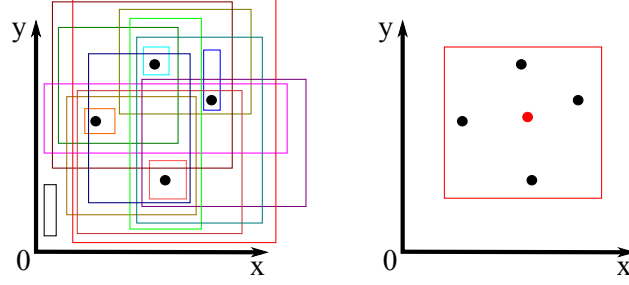


Figure 1: Example of range space and VC-dimension. The space of points is the plane \mathbb{R}^2 and the set of ranges is the set of all *axis-aligned rectangles*. The figure on the left shows graphically that it is possible to shatter a set of four points using 16 rectangles. On the right instead, one can see that it is impossible to shatter five points, as, for any choice of the five points, there will always be one (the red point in the figure) that is internal to the convex hull of the other four, so it would be impossible to find an axis-aligned rectangle containing the four points but not the internal one. Hence $\text{VC}((X, R)) = 4$.

R all the intervals $[a, b]$, such that $0 \leq a \leq b \leq 1$). Let $A = \{x, y, z\}$ be the set of three points $0 < x < y < z < 1$. No interval in R can define the subset $\{x, z\}$ so the VC-dimension of this range space is less than 3 [16, Lemma 10.3.1]. Another example is shown in Fig. 1.

The main application of VC-dimension in statistics and learning theory is its relation to the size of the sample needed to approximate learning the ranges, in the following sense.

Definition 8. Let (X, R) be a range space and let A be a finite subset of X . For $0 < \varepsilon < 1$, a subset $B \subset A$ is an ε -approximation for A if for all $r \in R$, we have

$$\left| \frac{|A \cap r|}{|A|} - \frac{|B \cap r|}{|B|} \right| \leq \varepsilon. \quad (2)$$

An ε -approximation can be constructed by random sampling points of the domain [11, Thm. 2.12, see also [14]].

Theorem 1. *There is an absolute positive constant c such that if (X, R) is a range-space of VC-dimension at most v , $A \subset X$ is a finite subset and $0 < \varepsilon, \delta < 1$, then a random subset $B \subset A$ of cardinality m , where*

$$m \geq \min \left\{ |A|, \frac{c}{\varepsilon^2} \left(v + \log \frac{1}{\delta} \right) \right\}, \quad (3)$$

is an ε -approximation for A with probability at least $1 - \delta$.

Note that throughout the work we assume the sample to be drawn *with* replacement if $m < |A|$, otherwise the sample is exactly the set A . The constant c is absolute and does not depend on the range space or on any other parameter. Löffler and Phillips [15] estimated experimentally that the absolute constant c is at most 0.5. Up to a constant, the bound presented in Thm. 1 is tight [14, Thm. 5].

It is also interesting to note that an ε -approximation of size $O(v\varepsilon^{-2}(\log v - \log \varepsilon))$ can be built *deterministically* in time $O(v^{3v}(\varepsilon^{-2}(\log v - \log \varepsilon))^v |X|)$ [5].

2.3 VC-Dimension and SQL Queries

Riondato et al. [20] defined a range space for SQL queries and computed a bound to its VC-Dimension. We recall here some of their results that we will use throughout the paper.

Let Q be a collection of (non-group-by, non-aggregate) queries. We define a range space $S_Q = (X, \mathcal{R})$ associated to Q as follows. The domain X is the *Cartesian product* of the tables involved in queries of Q . For each query $q \in Q$, let R_q be the subset of X in the output of q . The set \mathcal{R} is the collection of all the ranges R_q for each $q \in Q$. In the rest of the paper we will often identify a query q with its range R_q (i.e., with its output) and the set of range \mathcal{R} with

the class of queries Q . When the ranges represent all the possible outputs of queries in a class Q applied to database tables \mathcal{D} , the VC-dimension of the range space is the maximum number of tuples such that any subset of them is the output of a query in Q . The following lemmas show how to bound the VC-dimension of the range space of different classes of queries.


Lemma 1 ([20]). *Let \mathcal{T} be a table with m columns, let $b > 0$ and let $\Sigma_{\mathcal{T}}^{b*}$ be the set of selection queries on \mathcal{T} whose selection predicate is a Boolean combination of b clauses. Then, the VC-dimension of the range space $S_b = (\mathcal{T}, \Sigma_{\mathcal{T}}^{b*})$ is at most $3((m+1)b) \log((m+1)b)$.*

Lemma 2 ([20]). *Consider the class Q of queries that can be seen as combinations of select and joins on $u > 2$ tables $\mathcal{T}_1, \dots, \mathcal{T}_u$. Let $S_i = (\mathcal{T}_i, R_i)$, $i = 1, \dots, u$ be the range space associated with the select queries on the u tables. Let $v_i = VC(S_i)$. Let m be the maximum number of columns in a table \mathcal{T}_i . We assume $m \leq \sum_i v_i$.¹ Let $S_Q = (\mathcal{T}_1 \times \dots \times \mathcal{T}_u, R_Q)$ be the range space associated with the class Q . The range set R_Q is defined as follows. Let $\rho = (r_1, \dots, r_u)$, $r_i \in R_i$, and let ω be a sequence of $u-1$ join conditions representing a possible way to join the u tables \mathcal{T}_i , using the operators $\{>, <, \geq, \leq, =, \neq\}$. We define the range*

$$J_\rho^\omega = \{(t_1, \dots, t_u) : t_i \in r_i, \text{ s.t. } (t_1, \dots, t_u) \text{ satisfies } \omega\}.$$

R_Q is the set of all possible J_ρ^ω . Then,

$$VC(S_Q) \leq 4u \left(\sum_i VC(S_i) \right) \log(u \sum_i VC(S_i)).$$

 **NOTE:** We can probably remove the above two lemmas and just state the following.

The following Theorem can be obtained by combining the above results.

Theorem 2. *Let b be a positive integer parameter. Given a set $\mathcal{D} = \{\mathcal{T}_1, \dots, \mathcal{T}_u\}$ of u tables and a set \mathcal{C}_i of up to m columns for each table \mathcal{T}_i , consider the class $Q_{u,m,b}$ of all queries on \mathcal{D} with up to $u-1$ join and u select operations, where the part of the selection predicate involving table \mathcal{T}_i has conditions about some of the columns from \mathcal{C}_i and up to b Boolean operations, then*

$$VC((\mathcal{D}, Q_{u,m,b})) \leq 12u^2(m+1)b \log((m+1)b) \log(3u^2(m+1)b \log((m+1)b)).$$

It is clear from this Theorem that the bound to the VC-dimension depends only on the number of tables u , the number of columns m , and the number of joins b . We can then define the function

$$vc(u, m, b) = 12u^2(m+1)b \log((m+1)b) \log(3u^2(m+1)b \log((m+1)b)).$$

Riondato et al. [20] used Thm. 1 and the above bound to the VC-dimension of the range space $S = (X, Q)$ for a collection of queries Q to compute a sample size m such that a sample of size m of the database can be used to compute high-quality approximations of the selectivities of all queries in Q . Formally, if we let $\sigma(q)$ be the selectivity of a query $q \in Q$ when run on the original database \mathcal{D} and let $\tilde{\sigma}(q)$ be the selectivity of q when run on the sample, Riondato et al. [20] proved that

$$\Pr(\exists q \in Q \text{ s.t. } |\sigma(q) - \tilde{\sigma}(q)| \geq \varepsilon) \leq \delta,$$

for some ε and δ in $(0, 1)$.

 **To-Do:** Say something about how to create the sample.

In this present work, we leverage on this result to compute good approximations of the values of aggregate queries using a sample.

¹The assumption $m \leq \sum_i v_i$ is reasonable for any practical case.

2.4 Estimating the Selectivity of GROUP BY Queries

To warm up, we show how to use the results introduced above to compute the selectivity of group-by queries.

Let $\mathcal{D} = \mathcal{T}_1, \dots, \mathcal{T}_u$, and \mathcal{C}_i ($1 \leq i \leq u$), m , b , and $Q_{u,m,b}$ be as in Thm. 2. For a positive integer k , let $Q_{u,m,b}^k$ be the class of group-by queries $q(q^*, \mathcal{G})$ where $q^* \in Q_{u,m,b}$ and $\mathcal{G} = \{G_1, \dots, G_\ell\}$ is a subset of the columns in the tables of \mathcal{D} , under the condition that no more than k columns per table are in \mathcal{G} . We use the equivalence from Fact 1 to show how to compute an approximation of the sizes of the groups in the output of a group-by query.

Given \mathcal{G} , for any query $q \in Q_{u,m,b}^k$, let $E_{q,\mathcal{G}}$ be the set of queries with selection/join predicate as in (1). It is easy to see that all $q' \in E_{q,\mathcal{G}}$ are members of the class $Q_{u,p,b+2k}$, where $p = \max\{|\mathcal{C}_i \cap \mathcal{G}| : 1 \leq i \leq u\}$ ($p \leq m + k$). The class $Q_{u,p,b+2k}$ contains also other queries that are not members of any E_q . From Thm. 2, we have that $\text{VC}((\mathcal{D}, Q_{u,p,b+2k})) \leq \text{vc}(u, p, b + 2k)$.

Hence, if we build an ε -approximation \mathcal{S} for the range space $(\mathcal{D}, Q_{u,p,b+2k})$ (for example using Theorem 1) and execute queries from $Q_{u,m,b}^k$ on \mathcal{S} , then the selectivities on \mathcal{S} of queries from E_q will be within ε from their real selectivities. Thanks to the equivalence from Fact 1, this means that we can estimate the sizes of the groups in the output of a query $q \in Q_{u,m,b}^k$ by looking at the sizes of the groups in the output of q when run on \mathcal{S} .

📌 **NOTE:** I wonder whether the above should be more formal.

Note that our guarantees do not extend to the number of groups, i.e., we cannot guarantee that the number of groups in the output of the query when run on the sample is within ε from the number of groups in the output of the query when run on the large database.

3 Approximately Answering Aggregate Queries

The focus of this work is computing approximate answers to aggregate database queries by running them on a random sample of the original database. In this section we first describe our method to compute the *point estimation* (i.e., the estimation of the value of the query) and then show how to derive deterministic confidence intervals for our estimations by solving convex optimization problems.

3.1 Point Estimation

Given a database \mathcal{D} , let $q = (q^*, f)$ be an aggregate query with $f : \mathcal{D}(C_f) \times \mathbb{N}_0 \rightarrow \mathbb{R}$ for some column C_f of some table of \mathcal{D} (we used Fact 2 to define f). Let S be the set of pairs (v, c_v) where $v \in \mathcal{D}(C_f)$ and c_v is the number of times that v appears in the column C_f in the output of q^* when q^* is run on \mathcal{D} .

We approximate $f(S)$, i.e., the output of q , with $f(\tilde{S})$ where \tilde{S} is a collection of pairs (v, \tilde{c}_v) such that there is a pair $(v, c_v) \in S$ and $c_v - \gamma_v \leq \tilde{c}_v \leq c_v + \gamma_v$ where $0 \leq \gamma_v \leq \gamma$, for some fixed parameter γ . In particular, \tilde{c}_v is a function of 1. the number of times that v appears in the column C_f in the output of q^* when q^* is run on a sample, and 2. the size of the sample.

Let $Q_{u,m,b}$ be as in Thm. 2, and let $\mathcal{A} = Q_{u,m,b} \times \mathcal{F}$ [🟢 ← Is this the right expression?] be a class of aggregate queries (q^*, f) where $q^* \in Q_{u,m,b}$ and $f \in \mathcal{F}$, for some family \mathcal{F} of functions such that each $f \in \mathcal{F}$ goes from $\mathcal{D}(\mathcal{T}.C) \times \mathbb{Z}^+$ to \mathbb{R} , where $\mathcal{T}.C$ is a column in some table \mathcal{T} of \mathcal{D} . For any query $q^* \in Q_{u,m,b}$, let \mathcal{C}_{q^*} be its selection/join predicate. We can see the output of any $q^* \in Q_{u,m,b}$ as the union of the outputs of the queries whose selection/join predicate is \mathcal{C}_{q^*} AND'ed with a condition on the column C_f , i.e., whose selection/join predicate is in the form

$$\mathcal{C}_{q^*} \text{ AND } C_f = v; \quad (4)$$

where $v \in \mathcal{D}(C_f)$. Formally, let q_v^* be a query with selection predicate as in (4), for $v \in \mathcal{D}(C_f)$, and let $B_{q^*,f} = \{q_v^* : v \in \mathcal{D}(C_f)\}$. Then, the output of q^* is the union of the outputs of all queries in $B_{q^*,f}$.

Let now $A(Q_{u,m,b}, \mathcal{F})$ be the union of all $B_{q^*,f}$, for all $q^* \in Q_{u,m,b}$ and $f \in \mathcal{F}$. Consider the class $\bar{Q} = Q_{u,m,b} \cup A(Q_{u,m,b}, \mathcal{F})$. The range space (\mathcal{D}, \bar{Q}) has VC-dimension at most $\text{vc}(u, m + 1, b + 1)$ because \bar{Q} is a subset of the class $Q_{u,m+1,b+1}$. We can then build an ε -approximation \mathcal{S} for the range space (\mathcal{D}, \bar{Q}) and the selectivity $\tilde{\sigma}(\bar{Q})$

of each query $q \in \bar{Q}$ when run on \mathcal{S} will be within ε from its selectivity $\sigma(q)$ when run on \mathcal{D} . If we fix $\varepsilon, \delta \in (0, 1)$ and use Thm. 1 to compute the sample \mathcal{S} , we have

$$\Pr(\exists q \in \bar{Q} \text{ s.t. } |\tilde{\sigma}(q) - \sigma(q)| \geq \varepsilon) \leq \delta .$$

Let η be the ratio between the size of \mathcal{D} and the size of \mathcal{S} [✓ Explain better what it means.]. Given an aggregate query $q \in \mathcal{A}$, we will approximate $f(S)$ with $f(\tilde{S})$, where the members of \tilde{S} are the pairs $(v, \sigma_v \cdot \eta)$ with σ_v being the selectivity on \mathcal{S} of the query from $A(Q_{u,m,b}, \mathcal{F})$ corresponding to (q^*, f) and v , for all $v \in D(C)$. To compute this we do not need to run all such queries on \mathcal{S} and it suffices to run q on \mathcal{S} . Note that we are not yet using the relevant properties of ε -approximation (Def. 8), but the advantages of building such an \mathcal{S} will be evident after the following discussion.

📌 **NOTE:** It may look like the above discussion does not need to be that convoluted, and we may just say that we run the query on the sample. The problem is that the result of the query on the sample may have to be “scaled up” to the entire database. Think about sum.

✓ **TO-DO:** Make example with sum or average?

Properties of the estimation. When using a sample to compute an estimation for a quantity, it is highly desirable that the estimation has some properties that make it more informative about the real value of the quantity. One of the most desirable properties is the *absence of bias*.

✓ **TO-DO:** Describe how, in order to achieve this, one may need/want to compute a slightly different function for the aggregate, rather than the original. E.g. for variance and stddev.

Another relevant property for the estimation is *consistency*.

✓ **TO-DO:** Same as above.

3.2 Deterministic confidence intervals

Single point estimations for a parameter θ of interest, although useful on their own, become much more informative when accompanied with *confidence intervals*. These are intervals of the domain of θ and give information about the accuracy of the estimation (which falls within the interval) and the location of the real value of θ . Usually, confidence bounds are *probabilistic*, in the following sense. Given a quantity θ to be estimated, a $1 - \alpha$ confidence interval I is an interval of the domain of θ such that I contains θ with probability $1 - \alpha$. The probability $1 - \alpha$ should be interpreted in a frequentist framework. It represents the fraction of times that the interval (computed according to a fix procedure) contains the real value of quantity θ , over an infinite number of repetitions of the procedure to compute the interval. In our settings and given our goals [✓ which we never formally specified. . .], deriving probabilistic confidence bounds is not a viable option:

✓ **TO-DO:** Here we should explain that we cannot use these because we would have to apply a union bound over a very high number of queries, defeating the purpose.

. Our approach instead guarantees that, if the sample \mathcal{S} is an ε -approximation for the range space (\mathcal{D}, Q) (where Q is the class of queries that one may want to run), then the confidence interval we compute contains the real value of the query, for all aggregate queries built in Q . Another advantage of our method compared to probabilistic confidence intervals is that deriving a procedure to compute the latter is highly-specific on the aggregate function to be estimated (e.g., average rather than standard deviation), while our method is valid for *any* aggregate function. Haas [9] developed deterministic confidence intervals for aggregate queries, but our method computes intervals that are *uniformly more powerful* than those of [9] thanks to the fact that we can leverage the properties of ε -approximations.

✓ **TO-DO:** [9] has some tricks that may help us getting better estimations. Check those.

The method. Our method for computing deterministic confidence intervals exploits the properties of the ε -approximation. In particular, we use the fact that the selectivity on \mathcal{S} of all the queries in an appropriately defined class of queries, is close (within ε) from their real selectivity on \mathcal{D} .

Let $\mathcal{A} \dots$ [✓ Add the others] be as in Sect. 3.1.

Given and aggregate query $q = (q^*, f) \in$ [✓ In what?], let $V_{q^*}(C_f)$ be the set of values from $D(C_f)$ that appear in the column C_f in the output of q^* when run on \mathcal{D} . For now, assume that we know $V_{q^*}(C_f)$. We will remove this assumption later. Consider the set $B_{q^*,f} = \{q_v^* : v \in V_{q^*}(C_f)\}$. We want to find the sets $E^- = \{\varepsilon_v^- : v \in V_{q^*}(C_f)\}$, $E^+ = \{\varepsilon_v^+ : v \in V_{q^*}(C_f)\}$ such that the sets \tilde{S}^- and \tilde{S}^+ , whose members are respectively the pairs $(v, (\tilde{\sigma}(q_v^*) + \varepsilon_v^-) \cdot \eta)$ and $(v, (\tilde{\sigma}(q_v^*) + \varepsilon_v^+) \cdot \eta)$ for all $v \in V_{q^*}(C_f)$, are such that $f(\tilde{S}^-)$ minimizes and $f(\tilde{S}^+)$ maximizes $f(\cdot)$ among all possible choices of the sets E^- and E^+ which satisfy some constraints explained in the following which are derived from the fact that \mathcal{S} is a ε -approximation.

Since we assume we can build a total order relationship on $D(C_f)$, let v_1, \dots, v_ℓ be a labelling of the values in $V_{q^*}(C_f)$ in increasing sorted order. Let $q = (q^*, f)$, and let \mathcal{C} be the selection/join predicate of q^* . Let a, b be two values in $V_{q^*}(C_f)$. W.l.o.g. assume $a \leq b$. Consider the query $q_{a,b}^*$ whose selection predicate is

$$\mathcal{C} \text{ AND } C_f < a \text{ AND } C_f < b$$

✓ **To-do:** Is that \leq ?

Given that \mathcal{S} is a ε -approximation for (\mathcal{D}, Q) [✓ Fix Q], we have that the selectivity $\tilde{\sigma}(q_{a,b})$ of $q_{a,b}$ when run on \mathcal{S} is within ε from its selectivity $\sigma(q_{a,b})$ on \mathcal{D} . More precisely, if we let $c_{a,b}$ be the cardinality of $q_{a,b}$ on \mathcal{D} and $\tilde{c}_{a,b}$ the cardinality of $q_{a,b}$ on \mathcal{S} , we have that

$$\max\{\tilde{c}_{a,b}, (\tilde{\sigma}(q_{a,b}) - \varepsilon)\eta\} \leq c_{a,b} \leq \min\{1, (\tilde{\sigma}(q_{a,b}) + \varepsilon)\eta\}.$$

It should be evident that $\tilde{c}_{a,b} = \sum_{i=a}^b \tilde{c}_{i,i}$, and analogously for $c_{a,b}$, $\sigma(q_{a,b})$, and $\tilde{\sigma}(q_{a,b})$. This implies that the difference between [✓ complete.]

The optimization problem to solve to compute the sets E^- and E^+ is then the following (for E^- , and analogously for E^+):

- **Constants:** $0 \leq \varepsilon \leq 1, 0 \leq \tilde{\sigma}_v \leq 1, \forall v \in V_{q^*}(C_f)$.

- **Variables:** ε_v^-

- **Goal** minimize the function $f(\tilde{S}^-)$.

- **Constraints:**

- $\max\{0, \} \leq \sum_{\ell=i}^j \sigma.$
- $|\varepsilon_v^-| \leq \varepsilon, \forall v \in V_{q^*}(C_f).$
- $\left| \sum_{v \in V_{q^*}(C_f)} \varepsilon_v^- \right| \leq \varepsilon.$

The properties of the ε -approximation are used in the constraints. For all $v \in V_{q^*}(C_f)$, the selectivity σ_v is within ε from the real selectivity of the query with selection/join predicate as in (4), and this fact is reflected in the first constraint. The same is true for the query q^* and given that its selectivity is the sum of the σ_v , then the absolute of the sum of the possible errors for each of the σ_v must be bounded by ε , as in the second constraint.

Lemma 3. For any function f , let S , \tilde{S}^+ , and \tilde{S}^- be defined as above. Then, $f(\tilde{S}^-) \leq f(S) \leq f(\tilde{S}^+)$.

Proof. It follows from the properties of the ε -approximation that S is one of the sets whose elements satisfy the constraints in the optimization problems. Hence, $f(S)$ is considered as a possible solution to each optimization problems, and the thesis follows. \square

Until now, we assume to have perfect knowledge of $V_{q^*}(C_f)$, the set of values appearing in the column C_f in the output of q^* when the query is run on \mathcal{D} . When we run q^* on a sample (as we do to approximate the aggregate query), the output of q^* may only contain a subset \tilde{V}_{q^*} of the values and if we only considered the values in \tilde{V}_{q^*} , the obtained confidence bounds may not contain the real output. We therefore have to drop the assumption of knowing $V_{q^*}(C_f)$ and assume instead to know two values m and M , with $m \leq M$ such that m is a lower bound to the values in $V_{q^*}(C_f)$ and M is an upper bound to the values in V_q . We have $V_q \subseteq [m, M]$. The values m and M can be derived either from the selection predicate of q^* , if it involves the column C_f , or are usually available, for each column, in the set of statistics stored by the DMBS for query optimization [13]. Hass calls this the “system catalog”, I don’t know if that is correct.] Consider the optimization problem defined on $[m, M]$. We have

Lemma 4. *Every solution to the optimization problem on V_q is a solution to the optimization problem on $[m, M]$.*

Proof. By setting the variables in $[m, M] \setminus V_q$ to 0. \square

Corollary 1. *The solution to the maximization problem on $[m, M]$ is greater or equal to the solution to the maximization problem on V_q . Analogously for the minimization problems.*

In what follows we derive analytical bounds to the confidence bounds for some widely utilized functions. The formulas we obtain are a pessimization of the confidence bounds found by solving the optimization problem, in the sense that the confidence intervals obtained by using the formulas will be larger than the one suggested by the optimization problem. The bound we derive for SUM is of particular interest because we will use it to derive the other bounds.

Confidence bounds for AVERAGE. The function to be optimized would be

$$\frac{1}{\sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}|} \sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v.$$

This function is not linear, so instead of using it, we will use a linear function such that the confidence interval obtained by solving the optimization problems will be larger or equal than the one obtained by using the above. We will actually use two different functions, depending on whether we are maximizing or minimizing. In the first case, we use the function $\frac{1}{(\sigma_q - \varepsilon) |\mathcal{D}|} \sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v$. In the second case, the function will be $\frac{1}{(\sigma_q + \varepsilon) |\mathcal{D}|} \sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v$. It is easy to see that solving the respective optimization problems will give solutions that are worse (in an “interval width” sense) than the ones obtained using the original function. What is more, we now just need to optimize the second term of the product in each function (i.e., the sum), which we already did in the previous Paragraph. We then obtain the following results:

- **Upper Bound:** $\frac{1}{(\sigma_q - \varepsilon) |\mathcal{D}|} \Sigma_{m, M}^+$.
- **Lower Bound:** $\frac{1}{(\sigma_q + \varepsilon) |\mathcal{D}|} \Sigma_{m, M}^-$.

Confidence bounds for VARIANCE. The function to be optimized is

$$\frac{1}{(\sigma_q + \sum_{v \in V_q} \varepsilon_v) |\mathcal{D}|} \sum_{v \in V_q} (\sigma_q + \varepsilon_v) |\mathcal{D}| \left(v - \frac{1}{(\sigma_q + \sum_{w \in V_q} \varepsilon_w) |\mathcal{D}|} \sum_{w \in V_q} (\sigma_q + \varepsilon_w) |\mathcal{D}| w \right).$$

We use the following equivalence, which holds for any $n > 1$ and any sequence of i.i.d. random variables X_i (μ is the average of the X_i ’s).

$$\frac{1}{n-1} \sum_i (X_i - \mu)^2 = \frac{1}{n-1} \sum_i X_i^2 - \frac{1}{n^2 - n} \left(\sum_i X_i \right)^2.$$

Consider now the following pair of functions. It is easy to see that they are respectively greater or equal and smaller or equal to the original function, point by point (i.e., for the same setting of ε_v).

- **Maximization:**

$$\frac{1}{(\sigma_q - \varepsilon)|\mathcal{D}| - 1} \sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v^2 - \frac{1}{(\sigma_q - \varepsilon)^2 |\mathcal{D}|^2 - (\sigma_q + \varepsilon) |\mathcal{D}|} \left(\sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v \right)^2.$$

- **Minimization:**

$$\frac{1}{(\sigma_q + \varepsilon)|\mathcal{D}| - 1} \sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v^2 - \frac{1}{(\sigma_q + \varepsilon)^2 |\mathcal{D}|^2 - (\sigma_q - \varepsilon) |\mathcal{D}|} \left(\sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v \right)^2.$$

These functions are still not linear but we could bound them if we could find upper and lower bounds to $\sum_{v \in V_q} (\sigma_v + \varepsilon_v) |\mathcal{D}| v^2$. Such upper and lower bounds, which we denote as $\Psi_{m,M}^+$ and $\Psi_{m,M}^-$ can be found in a way similar to the one we followed for $\Sigma_{m,M}^+$ and $\Sigma_{m,M}^-$.

We can then derive the following bounds to the confidence interval for VARIANCE:

- **Upper Bound:**

$$\frac{1}{(\sigma_q - \varepsilon)|\mathcal{D}| - 1} \Psi_{m,M}^+ - \frac{1}{(\sigma_q - \varepsilon)^2 |\mathcal{D}|^2 - (\sigma_q + \varepsilon) |\mathcal{D}|} \times \Sigma_{a,b}^-.$$

- **Lower Bound:**

$$\frac{1}{(\sigma_q + \varepsilon)|\mathcal{D}| - 1} \Psi_{m,M}^- - \frac{1}{(\sigma_q + \varepsilon)^2 |\mathcal{D}|^2 - (\sigma_q - \varepsilon) |\mathcal{D}|} \Sigma_{a,b}^+.$$

Confidence bounds for STDEV.

✓ **To-DO:** Square root of the ones for VARIANCE? It looks like it.

Comparison with previous results In this Paragraph we compare our confidence bounds with the deterministic confidence bounds presented by Haas [10].

✓ **To-DO:** Fill.

Solving the optimization problems efficiently. The constraints are convex functions, therefore if f is a convex function, the problems are convex optimization problems and can be solved in polynomial time.

🚧 **NOTE:** It seems to be that the functions we are interested in are actually quadratic and the constraints define a convex polytope, so it is even better.

✓ **To-DO:** What else can we say here?

4 Experiments

✓ **To-DO:**

- Evaluation of the size estimation for group-by queries.
- Evaluation of the aggregate function estimation.
- Evaluation of the confidence bounds with comparison with the deterministic confidence bounds from [10].

5 Previous Work

Many of the early contributions and developments in the area of Approximate Query Processing (AQP) using sampling are presented in the survey by Das [6]. Here we focus on the contributions that are more related to ours.

Using workload information to improve the quality of AQP was explored by many authors [3, 4, 7, 12, 13]. The usual approach is to use this information to assign different weights to the tuples in the database and then use weighted sampling to ensure that tuples that are used more often have higher probability of appearing in the sample. Although we also develop our results for classes of queries representing the workload of the database, these classes can be completely defined by their SQL expression and are therefore completely data-independent, with the effect of making our results much more general and flexible.

Haas [9, 10] developed Hoeffding-like inequalities to derive accurate estimations of query answers and confidence bounds to them. Simultaneous statistical inference techniques like the union bound [17] must be used in order to derive uniform guarantees over a class of queries, which are known to be overly conservative when the number of queries is large. The work is nevertheless interesting because it presents a class of deterministic confidence intervals which can be compared with the ones we present in Sect. 3.

In the work by Chaudhuri et al. [4], the solution of an optimization problem suggests good strata into which partition the tuples, before using stratified sampling, in order to achieve higher quality in AQP. The obtained sample is such that the mean square error in estimating the selectivity of queries belonging to a given workload is minimized, but there is no quality guarantee on the maximum error. We instead solve optimization problems to find confidence bounds to our estimate of the query answer, giving a guarantee on the error we make.

Pol and Jermaine [19] use the bootstrap, a statistical tool, to derive confidence bounds to the estimates. In its simplest form, their work requires running the query multiple (possibly thousands) of times, but additional data structures allows for running the query only once. The major difference from our work is that the bootstrap gives *experimental* confidence bounds, while we give an *analytical* procedures and guarantees to derive the bounds.

The work of Xu et al. [24] also deals with developing confidence bounds to estimates of answers to queries involving GROUP BY conditions. They suggest that correlation between groups answers should be taken into account when computing the bounds. This can be done by solving an inference problem which can not be solved in polynomial time unless $P = NP$. The authors examine different heuristics to circumvent this limitation. In our work we do not need to take correlation between groups into account because the guarantees on the group size estimates hold uniformly and independently for all groups.

Rösch and Lehner [21] suggest a biased sample approach to improve the quality of estimates for small-sized groups. Their technique exploits the standard deviation of the aggregate values in a group to compute how many tuples from that group should be sampled. In order to do this, a fixed set of aggregate attributes must be specified. Our approach, on the other hand, is much more general in that we do not need this information, and aggregates can be computed on any column.

Agarwal et al. [1] present BlinkDB, a parallel engine for query processing that uses stratified sampling to approximate aggregate queries. BlinkDB exploits query workload information to select the best group of columns for the strata. The samples are selected depending on the query and on additional constraints on runtime and accuracy specified by the user. The system can not actually guarantee that the estimation has the required accuracy for all queries, as no correction for multiple queries (e.g. the union bound) is considered when computing sample size. On the contrary, our approach guarantees that, with probability at least $1 - \delta$, the confidence interval it compute contains the real value for all queries.

6 Conclusions

 **To-DO:** Fill.

References

- [1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.
- [2] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Hoboken, NJ, USA, third edition, 2008.
- [3] Surajit Chaudhuri, Gautam Das, Mayur Datar, Rajeev Motwani, and Vivek Narasayya. Overcoming limitations of sampling for aggregation queries. In *Proceedings of the 17th International Conference on Data Engineering, ICDE '01*, pages 534–542, 2001. doi: 10.1109/ICDE.2001.914867.
- [4] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32, June 2007. ISSN 0362-5915. doi: <http://doi.acm.org/10.1145/1242524.1242526>. URL <http://doi.acm.org/10.1145/1242524.1242526>.
- [5] Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-00357-1.
- [6] Gautam Das. Sampling methods in approximate query answering systems. In John Wang, editor, *Encyclopedia of Data Warehousing and Mining*, pages 1702–1707. Information Science Publishing, second edition, 2009.
- [7] Venkatesh Ganti, Mong-Li Lee, and Raghu Ramakrishnan. ICICLES: Self-tuning samples for approximate query answering. In *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00*, pages 176–187, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-715-3. URL <http://portal.acm.org/citation.cfm?id=645926.672017>.
- [8] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems - The Complete Book*. Prentice Hall, Upper Saddle River, NJ, USA, 2002. ISBN 978-0-13-187325-4.
- [9] Peter J. Haas. Hoeffding inequalities for join-selectivity estimation and online aggregation. Technical Report RJ 10040, IBM Almaden Research Center, San Jose, CA, USA, 1996.
- [10] Peter J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*, pages 51–62, August 1997. doi: 10.1109/SSDM.1997.621151.
- [11] Sarel Har-Peled and Micha Sharir. Relative (p, ε) -approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. ISSN 0179-5376. URL <http://dx.doi.org/10.1007/s00454-010-9248-1>.
- [12] Arnd Christian König and Gerhard Weikum. Combining histograms and parametric curve fitting for feedback-driven query result-size estimation. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, pages 423–434, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-615-7. URL <http://portal.acm.org/citation.cfm?id=645925.671668>.
- [13] Iosif Lazaridis and Sharad Mehrotra. Progressive approximate aggregate queries with a multi-resolution tree structure. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, SIGMOD '01*, pages 401–412, New York, NY, USA, 2001. ACM. ISBN 1-58113-332-4. doi: <http://doi.acm.org/10.1145/375663.375718>. URL <http://doi.acm.org/10.1145/375663.375718>.
- [14] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001. ISSN 0022-0000. doi: DOI:10.1006/jcss.2000.1741. URL <http://www.sciencedirect.com/science/article/B6WJ0-45B65MX-H/2/3b0f2dd50a30dcee8b995d701959f15f>.

- [15] Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 313–324. Springer-Verlag, 2009. URL http://dx.doi.org/10.1007/978-3-642-04128-0_29.
- [16] Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, Secaucus, NJ, USA, 2002. ISBN 0387953744.
- [17] Rupert G. Miller. *Simultaneous Statistical Inference*. Springer series in statistics. Springer-Verlag, New York, NY, USA, 1981.
- [18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.
- [19] Abhijit Pol and Christopher Jermaine. Relational confidence bounds are easy with the bootstrap. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 587–598, New York, NY, USA, 2005. ACM. ISBN 1-59593-060-4. doi: <http://doi.acm.org/10.1145/1066157.1066224>. URL <http://doi.acm.org/10.1145/1066157.1066224>.
- [20] Matteo Riondato, Mert Akdere, Uğur Çetintemel, Stanley B. Zdonik, and Eli Upfal. The VC-dimension of SQL queries and selectivity estimation through sampling. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part II*, volume 6912 of *Lecture Notes in Computer Science*, pages 661–676, Berlin / Heidelberg, 2011. Springer. ISBN 978-3-642-23782-9.
- [21] Philipp Rösch and Wolfgang Lehner. Sample synopses for approximate answering of group-by queries. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 403–414, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-422-5. doi: <http://doi.acm.org/10.1145/1516360.1516408>. URL <http://doi.acm.org/10.1145/1516360.1516408>.
- [22] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for engineering and information science. Springer-Verlag, New York, NY, USA, 1999. ISBN 9780387987804.
- [23] Vladimir N. Vapnik and Alexey J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.
- [24] Fei Xu, Christopher Jermaine, and Alin Dobra. Confidence bounds for sampling-based GROUP BY estimates. *ACM Trans. Database Syst.*, 33(3):16:1–16:44, September 2008. ISSN 0362-5915. doi: <http://doi.acm.org/10.1145/1386118.1386122>. URL <http://doi.acm.org/10.1145/1386118.1386122>.