The VC-Dimension of SQL Queries and Selectivity Estimation through Sampling[*]

Matteo Riondato[**], Mert Akdere, Uğur Çetintemel, Stanley B. Zdonik, and Eli Upfal

Department of Computer Science, Brown University, Providence, RI, USA
{matteo,makdere,ugur,sbz,eli}@cs.brown.edu

Abstract. We develop a novel method, based on the statistical concept of VC-dimension, for evaluating the selectivity (output cardinality) of SQL queries – a crucial step in optimizing the execution of large scale database and data-mining operations. The major theoretical contribution of this work, which is of independent interest, is an explicit bound on the VC-dimension of a range space defined by all possible outcomes of a collection (class) of queries. We prove that the VC-dimension is a function of the maximum number of Boolean operations in the selection predicate, and of the maximum number of select and join operations in any individual query in the collection, but it is neither a function of the number of queries in the collection nor of the size of the database. We develop a method based on this result: given a class of queries, it constructs a concise random sample of a database, such that with high probability the execution of *any* query in the class on the sample provides an accurate estimate for the selectivity of the query on the original large database. The error probability holds *simultaneously* for the selectivity estimates of *all* queries in the collection, thus the same sample can be used to evaluate the selectivity of multiple queries, and the sample needs to be refreshed only following major changes in the database. The sample representation computed by our method is typically sufficiently small to be stored in main memory. We present extensive experimental results, validating our theoretical analysis and demonstrating the advantage of our technique when compared to complex selectivity estimation techniques used in PostgreSQL and the Microsoft SQL Server.

1 Introduction

As advances in technology allow for the collection and storage of vast databases, there is a growing need for advanced machine learning techniques for speeding up the execution of queries on such large datasets. In this work we focus on the fundamental task of estimating the selectivity, or output size, of a database query, which is a crucial step in a number of query processing tasks such as execution plan optimization and resource allocation in parallel and distributed databases.

[*] Work was supported in part by NSF award IIS-0905553.
[**] Corresponding author.

The task of efficiently obtaining such accurate estimates has been extensively studied with solutions ranging from storage of pre-computed statistics on the tables' distribution, to on-line sampling of the databases, and to combinations of the two approaches [39, 40, 24, 30, 25, 15, 16, 20, 31, 37, 49]. Histograms, simple yet powerful statistics of the tables' data, are the most commonly used solution in practice, due to their computational and space efficiency. However, there is an inherent limitation to the accuracy of this approach when estimating the selectivity of queries that involve either multiple tables/columns or correlated data. Running the query on freshly sampled data gives more accurate estimates at the cost of delaying the execution of the query while collecting random samples from a disk or other large storage medium and performing the analysis itself, which is usually more expensive than a histogram lookup. Our goal in this work is to leverage the computational efficiency of using pre-collected data with the provable accuracy of estimates obtained by running a query on a properly selected sample database.

We apply the statistical concept of VC-dimension [53] to develop and analyze a novel technique for generating accurate estimates of query selectivity. Roughly speaking, the VC-dimension of a collection of indicator functions (hypotheses) is a measure of its complexity or expressiveness (see Sect. 3 for formal definitions). A major theoretical contribution of this work, which is of independent interest, is an explicit bound to the VC-dimension of a class of queries, viewed as indicator functions on the Cartesian product of the database tables. In particular, we show that the VC-dimension of a class of queries is a function of the maximum number of Boolean, select and join operations in any query in the class, but it is not a function of the number of different queries in the class. By adapting a fundamental result from the VC-dimension theory to the database setting, we develop a method that for any class of queries, defined by its VC-dimension, constructs a concise sample of a database, such that with high probability, the execution of *any* query in the class on the sample provides an accurate estimate for the selectivity of the query on the original large database. The error probability holds *simultaneously* for the selectivity estimate of *all* queries in the collection, thus the same sample can be used to evaluate the selectivity of multiple queries, and the sample needs to be refreshed only following major changes in the database. The size of the sample does not depend on the size (number of tuples) in the database, just on the complexity of the class of queries we plan to run, measured by its VC-dimension. Both the analysis and the experimental results show that accurate selectivity estimates are obtained using a surprising small sample size (see Table 1 for concrete values), which allows the entire sample to reside in main memory, significantly speeding up the execution of the query on the sample.

A technical difficulty in applying the VC-dimension approach to the database setting is that the VC-dimension analysis assumes a uniform sample of the Cartesian products of all the tables, while in practice, it is more efficient to run the queries on the Cartesian product of random samples of the individual tables (which has a different distribution). We develop an efficient procedure for constructing a sample that circumvents this problem (see Sect. 5).

We present extensive experimental results, validating our theoretical analysis and demonstrating the advantage of our technique when compared to complex selectivity estimation techniques used in PostgreSQL and the Microsoft SQL Server. The main advantage of our methods is that it gives provably accurate prediction for all queries with up to a given complexity (VC-dimension), while techniques like multidimensional histograms or join synopses are accurate only for the queries for which they are built.

Note that we are only concerned with estimating the selectivity of a query, not with approximating the query answer using a sample of the database (see the work by Das [12] for a survey of that area).

Due to space limitation we focus here on the main novel concepts of our method. Full proofs and additional experimental results are included in the full paper [51].

2 Related Work

Methods for estimating the selectivity (or cardinality) of queries have been extensively studied in the database literature. A variety of approaches have been explored, ranging from the use of sampling, both online and offline, to precomputation of different statistics such as histograms, to building on methods from machine learning [11,28], data mining [21], optimization [7,42], and probabilistic modeling [19,50].

The use of sampling for selectivity estimation has been studied mainly in the context of online sampling [40,39], where the sample is obtained after a query has arrived and only used for the evaluation of the selectivity of that one query. Sampling at random from a large database residing on disk is an expensive operation [47,4,18], and in some cases sampling for an accurate cardinality estimate is not significantly faster than full execution of the query [22,23]. A variety of sampling and statistical analysis techniques has been tested for improving the efficiency of the sampling procedures and in particular identifying early stopping conditions [30,24,25,15,56,3,14,7,35] but online sampling is still considered too expensive for most applications. An offline sampling approach was explored by Ngu et al. [46] who used systematic sampling (requiring the tuples in a table to be sorted according to one of the attributes) with a sample size dependent on the number of tuples in the table. The paper does not give any explicit guarantee on the accuracy of their predictions. Chaudhuri et al. [7] present an approach which uses optimization techniques to identify suitable strata before sampling. 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 error for each of the queries. Haas [27] developed Hoeffding inequalities to bound the probability that the selectivity of a query estimated from a sample deviates more than a given amount from its expectation. However, to estimate the selectivity for multiple queries and obtain a given level accuracy for all of them, simultaneous statistical inference techniques like the union bound should be used, which are known to be overly conservative when the number of queries is large [45]. On the contrary, our result will hold simultaneously for *all* queries within a given complexity (VC dimension).

A technical problem arises when combining join operations and sampling, as pointed out by Chaudhuri et al. [9]: the Cartesian product of the samples of two tables is not a uniform sample of the Cartesian product of the tables. What is more, given a size s, it is impossible a priori to determine two sample sizes s_1 and s_2 such that samples of these sizes from the two tables will give, when joined together along a common column, a sample of the join table of size s. In Sect. 5 we explain why only the first issue is of concern for us and how we circumvent it.

In practice most database systems use pre-computed statistics to predict query selectivity [31, 20, 16, 34, 37], with histograms being the most commonly used representation. The construction, maintenance, and use of histograms were thoroughly examined in the literature [33, 32, 43, 48], with both theoretical and experimental results. In particular Chaudhuri et al. [8] rigorously evaluated the size of the sample needed for building a histogram providing good estimates for the selectivities of a large group of (select only, in their case) queries. Kaushik et al. [36] extensively compared histograms and sampling from a space complexity point of view, although their sample-based estimator did not offer a uniform probabilistic guarantee over a set of queries and they only consider the case of foreign-key equijoins. We address both these points in our work. Although very efficient in terms of storage needs and query time, the quality of estimates through histograms is inherently limited for complex queries by two major drawbacks: intra-bucket uniformity assumption (i.e., assuming a uniform distribution for the frequencies of values in the same bucket) and inter-column independence assumption (i.e., assuming no correlation between the values in different columns of the same or of different tables). Different authors suggested solutions to improve the estimation of selectivity without making the above assumptions [5, 13, 49, 55, 54]. Among these solutions, the use of multidimensional histograms [6, 49, 52, 54] seems the most practical. Nevertheless, these techniques are not widespread due to the extra memory and computational costs in their implementation.

To the best of our knowledge, our work is the first to provide explicit bounds on the VC-dimension of queries and to apply the results to query selectivity estimation.

3 Preliminaries

We consider a database $\mathcal{D}$ of k tables $\mathcal{T}_1, \ldots, \mathcal{T}_k$. 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 $t.C$. The values in $\mathcal{T}.C$ belong to the numerical or categorical domain $D(\mathcal{T}.C)$. Our focus is on queries that combine select and join operations, defined as follows. We do not take projection operations into consideration because their selectivities have no impact on query optimization.

Definition 1. *Given a table $\mathcal{T}$ with columns $\mathcal{T}.C_1, \ldots, \mathcal{T}.C_\ell$, 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.*

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 $\{(t_1, t_2) \ : \ t_1 \in \mathcal{T}_1, t_2 \in \mathcal{T}_2, \text{ s.t. } t_1.C \,\mathrm{op}\, t_2.C\}$ where "op" is one of $\{<, >, \geq, \leq, =, \neq\}$.*

Definition 3. *Given a set of ℓ tables $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$, a* combination of select and join operations *is a query returning a subset of the Cartesian product of the tuples in the sets $S_1, \ldots, S_\ell$, 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 $S_1, \ldots, S_\ell$.*

Definition 4. *Given a query q, a* query execution plan *for q is a directed binary tree T_q whose nodes are the elementary operations (i.e. select or join queries) into which q can be decomposed. There is an edge from node A to node B if the output of A is used as an input to B.*

It follows from the definition of a combination of select and join operations that a query may have multiple execution plans. Nevertheless, for all the queries we defined there is (at least) one execution plan such that all select operations are in the leaves and internal nodes are join nodes [17]. To derive our results, we use these specific plans.

Two crucial definitions that we use throughout the work are the *cardinality* of the output of a query and the equivalent concept of *selectivity* of a query.

Definition 5. *Given a query q and a database $\mathcal{D}$, 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, when run on $\mathcal{D}$. The* selectivity *$\sigma(q)$ of q is the ratio between its cardinality and the product of the sizes of its input tables.*

VC-Dimension. The Vapnik-Chernovenkis (VC) Dimension of a space is a measure of complexity or expressiveness of a set of functions on that space [53]. A finite bound on the VC-dimension of a structure implies a bound on the size of random samples required for approximately learning that structure. We outline some basic definitions and results and their adaptation to the specific setting of queries. We refer the reader to the works of Alon and Spencer [2, Sect.14.4], and Chazelle [10, Chap. 4] for an in-depth discussion of the VC-dimension theory. VC-dimension is defined on *range spaces*:

Definition 6. *A* range space *is 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*.*

In our setting, for a class of select queries Q on a table $\mathcal{T}$, X is the set of all tuples in the input table, and R the family of the outputs of the queries in Q when run on X, i.e. on $\mathcal{T}$. For a class Q of queries combining select and join operations, X is the Cartesian product of the associated tables and R is the family of outcomes of queries in Q, seen as ℓ-uples of tuples, if ℓ tables are involved in the queries of Q. When the context is clear we identify the family R with a class of queries.

Definition 7. *Let (X, R) be a range space and $A \subset X$. The* projection *of R on A is defined as $P_R(A) = \{r \cap A \;:\; r \in R\}$.*

Definition 8. *Let (X, R) be a range space and $A \subset X$. If $|P_R(A)| = 2^A$, then A is said to be* shattered by *R.*

Definition 9. *Let $S = (X, R)$ be a range space. The* Vapnik-Chervonenkis *dimension (or* VC-dimension*) of S, denoted as $VC(S)$ is the maximum cardinality of a shattered subset of X. If there are arbitrary large shattered subsets, then $VC(S) = \infty$.*

When the range space represents all the possible outputs of queries in 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 defined by a query in Q.

The main application of VC-dimension in statistics and learning theory is its relation to the minimum sample size needed for approximate learning of a function on the point space using a range.

Definition 10. *Let (X, R) be a range space and let A be a finite subset of X.*

1. *For $0 < \varepsilon < 1$, a subset $B \subset A$ is an ε-*approximation *for A if for any range $r \in R$, we have $\left|\frac{|A \cap r|}{|A|} - \frac{|B \cap r|}{|B|}\right| \leq \varepsilon$.*
2. *For $0 < p, \varepsilon < 1$, a subset $B \subset A$ is a* relative (p, ε)-approximation *for A if for any range $r \in R$ such that $\frac{|A \cap r|}{|A|} \geq p$ we have $\left|\frac{|A \cap r|}{|A|} - \frac{|B \cap r|}{|B|}\right| \leq \varepsilon \frac{|A \cap r|}{|A|}$ and for any range $r \in R$ such that $\frac{|A \cap r|}{|A|} < p$ we have $\frac{|B \cap r|}{|B|} \leq (1 + \varepsilon)p$.*

It is possible to probabilistically build an ε-approximation (resp. a relative (p, ε)-approximation) by sampling the point space [53, 38, 29].

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

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

(resp. $m \geq \min\left\{|A|, \frac{c'}{\varepsilon^2 p}\left(d \log\frac{1}{p} + \log\frac{1}{\delta}\right)\right\}$) is an ε-approximation (resp. a relative (p, ε)-approximation) for A with probability at least $1 - \delta$.

Löffler and Phillips [41] showed experimentally that the constant c is approximately 0.5. It is also interesting to note that an ε-approximation of size $O(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon})$ can be built *deterministically* in time $O(d^{3d}(\frac{1}{\varepsilon^2} \log \frac{d}{\varepsilon})^d |X|)$ [10].

4 The VC-Dimension of Classes of Queries

In this section we develop a general bound on the VC-dimension of classes of queries. We start by computing the VC-dimension of simple select queries and move to queries with complex selection predicates and to join queries. We then extend our bounds to general queries that are combinations of multiple select and join operations. The proofs for our results can be found in the full version of this paper [51].

Select Queries. Let $\mathcal{T}$ be a table with m columns $\mathcal{T}.C_1, \ldots, \mathcal{T}.C_m$. For a fixed column $\mathcal{T}.C_i$, consider the family $\Sigma^*_{C_i}$ of all possible outputs of queries in the form

$$\texttt{SELECT * FROM } \mathcal{T} \texttt{ WHERE } \mathcal{T}.C_i \,\mathrm{op}\, a \tag{2}$$

where op is an inequality operator (i.e., either "$\geq$" or "$\leq$")[1] and $a \in D(\mathcal{T}.C_i)$.

Lemma 1. *Let $\mathcal{T}$ be a table with m columns C_i, $1 \leq i \leq m$, and consider the set of queries $\Sigma^*_{\mathcal{T}} = \bigcup_{i=1}^m \Sigma^*_{C_i}$, where $\Sigma^*_{C_i}$ is defined as in the previous paragraph. Then, the range space $S = (\mathcal{T}, \Sigma^*_{\mathcal{T}})$ has VC-dimension at most $m + 1$.*

This lemma follows easily from a well known result on the VC-dimension of half-spaces in $\mathbb{R}^m$ [44, Lemma 10.3.1].

We now extend the bound to general selection queries with complex predicates. Given a table $\mathcal{T}$ with m columns, consider the set $\Sigma^{b*}_{\mathcal{T}}$ of queries whose selection predicate can be expressed as the set of selection queries whose predicate is a Boolean combination of at most b clauses, i.e. the query is in the form

$$\texttt{SELECT * FROM } \mathcal{T} \texttt{ WHERE } \mathcal{T}.C_{i_1}\, \mathrm{op}_1\, a_1\, \mathrm{bool}_1 \cdots \mathrm{bool}_{b-1}\, \mathcal{T}.C_{i_b}\, \mathrm{op}_b\, a_b$$

where op_i is one of "$\geq$", "$\leq$", "bool_ℓ" is either AND or OR, $i_j \in [1, m]$, and $a_j \in D(\mathcal{T}.C_{i_j})$, $1 \leq j \leq b$. It should be noted that the clauses in the selection predicate may be parenthesized in many different ways, each resulting (potentially) in a different query. All the possible parenthesizations are members of the range space $\Sigma^{b*}_{\mathcal{T}}$. It is also important to realize that we can and we do see a selection clause involving the "$=$" operator or the "$\neq$" operator as the "AND" of two clauses involving the $>$ and $<$ operators.

Lemma 2. *Let $\mathcal{T}$ be a table with m columns, let $b > 0$ and let $\Sigma^{b*}_{\mathcal{T}}$ be the set of selection queries on $\mathcal{T}$ whose selection predicate is a Boolean combination of up to b clauses. Then, the VC-dimension of the range space $S_b = (\mathcal{T}, \Sigma^{b*}_{\mathcal{T}})$ is at most $3((m + 1)b)\log((m + 1)b)$.*

Note that not all queries in $\Sigma^{b*}_{\mathcal{T}}$ are equivalent to axis-aligned boxes, thus we can not apply the bound used in the proof of Lemma 1. Instead, we use the following extension of [2, Corol. 14.4.3] to arbitrary combinations of set operations.

[1] The operators "$>$" and "$<$" can be reduced to "$\geq$" and "$\leq$" respectively.

Lemma 3. *Let (X, R) be a range space of VC-dimension $d \geq 2$ and let (X, R_h) be the range space on X in which R_h include all possible combinations of union and intersections of h members of R. Then $VC(X, R_h) \leq 3dh \log(dh)$.*

To prove Lemma 2 using Lemma 3, we observe that the outcome of an `AND` (resp. `OR`) operator connecting two selection clauses is equal to the intersection (resp. union) of the two selection clauses outputs.

Join Queries. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two distinct tables, and let R_1 and R_2 be families of (outputs of) queries on the tuples of $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. Let $S_1 = (\mathcal{T}_1, R_1)$, $S_2 = (\mathcal{T}_2, R_2)$ and let $VC(S_1), VC(S_2) \geq 2$. Let C be a column along which $\mathcal{T}_1$ and $\mathcal{T}_2$ are joined, and let $T_J = \mathcal{T}_1 \times \mathcal{T}_2$ be the Cartesian product of the two tables. For a pair of queries $r_1 \in R_1$, $r_2 \in R_2$, let

$$J^{\mathrm{op}}_{r_1,r_2} = \{(t_1, t_2) \ : \ t_1 \in r_1, t_2 \in r_2, t_1.C \,\mathrm{op}\, t_2.C\},$$

where op $\in \{>, <, \geq, \leq, =, \neq\}$. We have $J^{\mathrm{op}}_{r_1,r_2} \subseteq r_1 \times r_2$ and $J^{\mathrm{op}}_{r_1,r_2} \subseteq T_J$. Let $J_C = \{J^{\mathrm{op}}_{r_1,r_2} \mid r_1 \in R_1, r_2, R_2, \mathrm{op} \in \{>, <, \geq, \leq, =, \neq\}\}$. We have $VC((T_J, J_C)) \leq 3(VC(S_1) + VC(S_2)) \log((VC(S_1) + VC(S_2)))$. This result can be extended to queries with multiple joins operations:

Lemma 4. *Consider the class Q of queries that can be seen as combinations of select and join operations on $u > 2$ tables $\mathcal{T}_1, \ldots, \mathcal{T}_u$. Let $S_i = (\mathcal{T}_i, R_i)$, $i = 1, \ldots, u$ 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 $m \leq \sum_i v_i$.*[2] *Let $S_Q = (\mathcal{T}_1 \times \cdots \times \mathcal{T}_u, R_Q)$ be the range space associated with the class Q. The range set R_Q is defined as follows. Let $\rho = (r_1, \ldots, r_u)$, $r_i \in R_i$, 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*

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

R_Q is the set of all possible J^{ω}_{ρ}. Then,

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

We could not reduce the claim of this lemma to any know result in the VC-dimension theory. Our proof constructs a bound on the maximum size of a shattered set, by considering the effect of the different operators on the output. See the full version of the paper [51] for more details.

General Queries. Combining the above results we prove:

Theorem 2. *Consider the class $Q_{u,m,b}$ of all queries with up to $u - 1$ join and u select operations, where each select operation involves no more than m columns and b Boolean operations, then $VC(Q_{u,m,b}) \leq 12u^2(m+1)b \log((m+1)b) \log(3u^2(m+1)b \log((m+1)b))$.*

[2] The assumption $m \leq \sum_i v_i$ is reasonable for any practical case.

5 Implementation

Consider a database $\mathcal{D}$ and a class of queries $Q_{u,m,b}$. Theorem 2 gives a bound to the VC-dimension of the range space $(\mathbf{D}, Q_{u,m,b})$, where $\mathbf{D}$ is the Cartesian product of all the tables in $\mathcal{D}$. Theorem 1 gives the required size of a uniform random sample $\mathcal{S}$ of $\mathbf{D}$, such that the execution of *any* query $q \in Q_{u,m,b}$ on $\mathcal{S}$ gives an ε-approximation (or a relative (p, ε)-approximation) of the selectivity of q when executed on $\mathcal{D}$ (see Table 1 for concrete values). Note that for any execution plan of a query $q \in Q_{u,m,b}$, all the queries that correspond to subtrees rooted at internal nodes of the plan are queries in $Q_{u,m,b}$. Thus, by running query q on the sample we obtain accurate estimates for the selectivity of all the subqueries defined by its execution plan.

In practice, it is more efficient to maintain the table structure of the original database in the sample. It is easier to sample each table independently, and to run the query on a sample that consists of subsets of the original tables rather than re-writing the query to run on a Cartesian product of tuples. However, the Cartesian product of independent uniform samples of tables is not a uniform sample of the Cartesian product of the tables [9]. We developed a procedure to circumvent this problem. Due to space constraints, we present here an informal description of the procedure and refer the interested reader to the full version of the paper [51]. Assume that we need a uniform sample of size t from $\mathbf{D}$, which is the Cartesian product of ℓ tables $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$. We then sample t tuples uniformly at random from each table $\mathcal{T}_i$, to form a sample table $\mathcal{S}_i$. We add an attribute *sampleindex* to each $\mathcal{S}_i$ and we set the value in the added attribute for each tuple in $\mathcal{S}_i$ to a unique value in $[1, t]$. Now, each sample table will contain t tuples, each tuple with a different index value in $[1, t]$. Given an index value $i \in [1, t]$, consider the set of tuples $X_i = \{x_1, \ldots, x_\ell\}$, $x_j \in \mathcal{S}_i$ such that $x_1.\mathit{sampleindex} = x_2.\mathit{sampleindex} = \cdots = x_\ell.\mathit{sampleindex} = i$. X_i can be seen as a tuple sampled from $\mathbf{D}$, and the set of all X_i, $i \in [1, t]$ is a uniform random sample of size t from $\mathbf{D}$. We run queries on the sample tables, but in order to estimate the selectivity of a join operation we count a tuple Y in the result only if the set of tuples composing Y is a subset of X_i for some $i \in [1, t]$. This is easily done by scanning the results and checking the values in the *sampleindex* columns.

Note that our method circumvent the major difficulty pointed out in [9]. They proved that, in general, it is impossible to predict sample sizes for given two tables such that the join of the samples of two tables will result in a sample of a required size out of the join of the two tables. Our method does not require a sample of a given size from the result of a join. The VC-dimension sampling technique requires only a sample of a given size from the Cartesian product of the tables, which is guaranteed by the above procedure.

6 Experiments

The first goal of the experiments is to evaluate the practical usefulness of our theoretical results. To assess this, we run queries on a large database and on

Table 1. Sample Sizes (tuples)

		Select		Join	
m	b	VC-dim	Sample size	VC-dim	Sample size
1	1	2	1000	4	1400
	2	4	1400	16	3800
	3	6	2800	36	7800
	5	10	2600	100	20600
	8	16	3800	256	51800
2	2	31	6800		
	3	57	12000		
	5	117	24000		
	8	220	44600		
5	5	294	59400		

sample representations generated by our method.[3] We used the selectivity of the each query in the random samples as an estimator for the selectivity in the large database (with the appropriate adjustments for join operations, as described in the previous section). We computed the error between the estimate and the actual selectivity to validate the analysis in Thm. 1 in practical settings. The use of a large number of queries and a variety of parameters allowed us to evaluate the error a function of the sample size. Our second goal is comparing our results, which give probabilistic guarantees on the error of the predicted selectivity, with the standard selectivity estimation done using precomputed statistics as implemented in PostgreSQL and Microsoft SQL Server. Additional experimental results can be found in the full version of the paper [51].

Setup. The tables in our large database were randomly generated and contain 20 million tuples. The distributions of values in the columns fell in two different categories: **1.** *uniform and independent*: the values in the columns were chosen uniformly and independently at random from a fixed domain. Each column was treated independently from the others. Tables involved in join queries belonged to this category only. **2.** *correlated*: two columns of the tables contained values from a multivariate normal distribution with mean $M = \mu\mathbb{I}_{2,2}$ and a non-identity covariance matrix Σ. We sampled tuples from the large tables uniformly, independently, and with replacement, to build the sample tables. For the samples of the tables used to run join queries, we added a column *sampleindex* to each tuple as described in Sect. 5. For each table in the original database we created many sample tables of different sizes, either fixed arbitrarily to 1000, 2000, or 5000 tuples or computed using (1). To compute the VC-dimension-dependent sample size, we fixed $\varepsilon = 0.05$, $\delta = 0.05$, and $c = 0.5$, as suggested by Löffler and Phillips [41]. The parameter d was set to the best bound to the VC-dimension of the range space of the queries we were running, as obtained from our theoretical results. We used $m = 1, 2$ (only $m = 1$ for joins) and $b = 1, 2, 3, 5, 8$, with the

[3] We focused on building ε-approximations, but relative (p, e)-approximations would give similar results.

addition of the combination $m = 5$, $b = 5$. Table 1 shows the sample sizes as number of tuples. We stress again the fact that the sample sizes are independent from the sizes of the original tables, so the larger the original table, the smaller will be the ratio between the sample size and the original size and the higher the gains in terms of space. We built PostgreSQL histograms with a different number of buckets, ranging from 100 to 10000. For SQL Server, we built the standard single-column histograms and computed the multi-column statistics which should help obtaining better estimations when the values along the columns are correlated. For each combination of the parameters m and b and each large table (or pair of large tables, in the case of join) we created 100 queries, with selection predicates chosen independently and uniformly at random, involving m columns and b Boolean clauses.

Results. A major result of our experiments is that for all the queries we run and all the sample tables the estimate of the selectivity computed using the selectivity in the sample was within ε (0.05) from the real selectivity. The same was not true for the selectivity computed by the histograms. As an example, in the case of $m = 2$, $b = 5$ and uniform independent columns, the default PostgreSQL histograms predicted a selectivity more than ε off from the real selectivity for 30 out of 100 queries. Nevertheless, from time to time the histograms predicted a selectivity closer to the actual one than the prediction from the sample. This is especially true when the histogram assumption are verified (e.g., for $m = 1, b = 1$ the default PostgreSQL histograms gave a better prediction than the sample in 28 out of 100 cases). This "inversion of precision" becomes less and less frequent as the sample size grows and as the complexity of the queries grows. Since the selectivity estimated by the sample was *always* within ε from the actual, we focused on the percent error, i.e. on the quantity $e_{\%} = \frac{100|p(\sigma_q)-\sigma_{\mathcal{D}}(q)|}{\sigma_{\mathcal{D}}(q)}$ where $p(\sigma_q)$ is the predicted selectivity. We can see from Fig. 1 and 2 that both the average and the standard deviation of the percentage error of the sample prediction decrease as the sample size grows. Much more interesting than this is the comparison between the performance of the histograms and the performance of the sample in predicting selectivities. When the assumptions of the histograms hold, as is the case for the data plotted in Fig. 1, the predictions obtained from the histograms can be of good quality. As said though, for a majority of queries, the prediction from the sample is better than the one from the histograms.

But as soon as the data are correlated (Fig. 2), the sample gives better predictions than the histograms even at the smallest sample sizes and keeps improving as the sample grows larger. In Fig. 2 we do not show multiple curves for the different PostgreSQL histograms because increasing the number of buckets had very marginal impact on the quality of the estimates, sometime even in the negative sense (i.e., an histogram with more buckets gave worse predictions than an histogram with less buckets), a fact that can be explained with the variance introduced by the sampling process used to create the histograms. For the same reason we do not plot multiple lines for the prediction obtained from the multi-columns and single-column statistics of SQL Server. The strengths of our method compared to histograms become more evident when we run join queries.

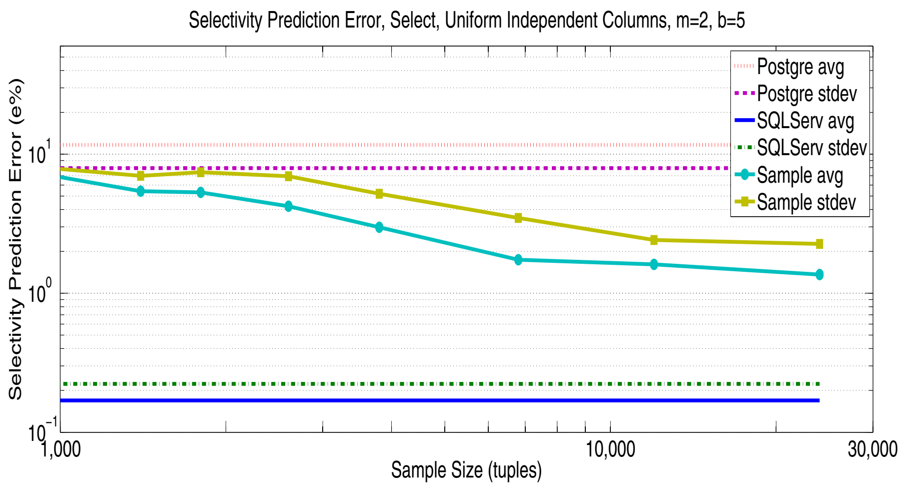

Fig. 1. Select – Uniform Independent Columns – $m = 2$, $b = 5$

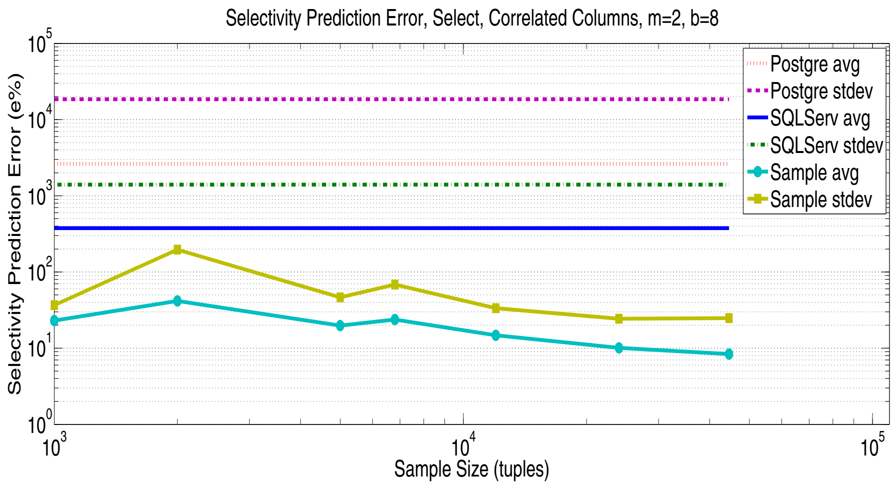

Fig. 2. Select – Correlated Columns – $m = 2$, $b = 8$

In our experiments, the predictions obtained from the sample were always within ε from the real values, even at the smallest sample sizes, but the same was not true for histograms. Figure 3 shows the comparison between the average and the standard deviation of the percentage error for the histograms and the sample. The numbers include predictions for the selection operations at the leaves of the query tree. The extremely bad performances of PostgreSQL is due to the fact that for some join queries, the histograms may predict an output size on the order of the hundreds of thousands tuples but the actual output size was zero or a very small number of tuples. Such errors drive the average and the standard

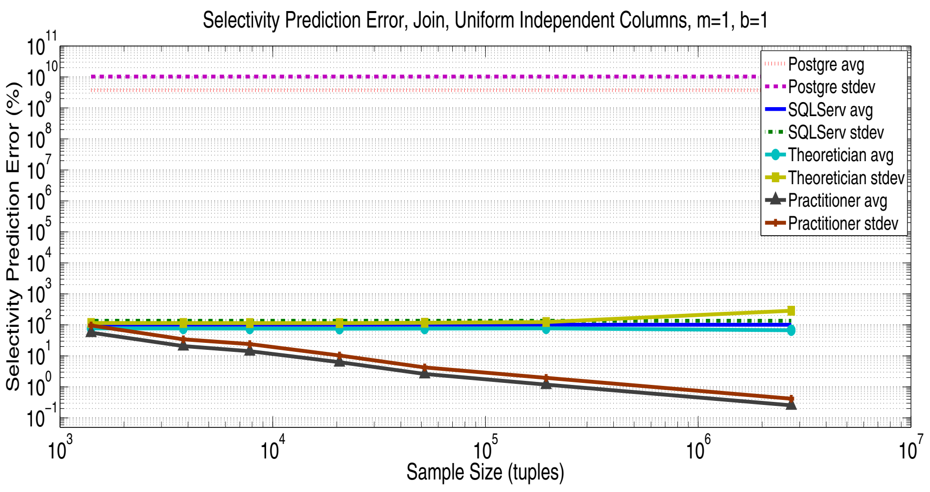

Fig. 3. Join – Uniform Independent Columns – $m = 1$, $b = 1$

deviation to very high values, but the comparison with the sample is fair and the prediction from the histograms are just not of good quality in such cases. The performance of SQL Server can be explained by the fact that this DBMS does not only use histograms to predict the selectivity of the query but also analyzes the query predicates and its query optimizer is very good in understanding when a query would return an empty output, therefore avoiding major errors in the estimation. This is something that vanilla histograms could not do, so the comparison with the sample is actually a bit unfair against the sample. Figure 3 also shows a comparison between the percentage error of predictions obtained from the sample in two different ways: the "theoretically correct" way that makes use of the number of pairs of tuples with the same value in the *sampleindex* column to predict the selectivity and the "practitioner" way which uses the size of the output of the join operation in the sample, ignoring the *sampleindex* column, i.e., without filtering out the tuples not belonging to the sample of the Cartesian product of the original tables.

From the experiments we ran we can conclude that our method for estimating the selectivity is a viable option in practice. The theoretical guarantees were always satisfied, with a consistency and an accuracy even higher than guaranteed. This fact can be explained by the potential looseness of the bounds to the VC-dimension of queries, and therefore to the sample size. From a practical point of view, it is also interesting that a sample of the Cartesian product is not necessary, and a certain level of non-uniformity in the sampling process may be accommodated. It may even well be that it is not necessary to use uniform independent sampling in order to obtain a ε-approximation.

7 Conclusions

We develop a novel method for estimating the selectivity of queries by executing it on a concise, properly selected, sample of the database. We present a rigorous analysis of our method and extensive experimental results demonstrating its efficiency and the accuracy of its predictions.

Most commercial databases use histograms built on a single column, for selectivity estimation. There has also been significant research on improving the estimate using multidimensional histograms [6, 49, 52, 54] and join synopses [1]. The main advantage of our method is that it gives uniformly accurate estimate for the selectivity of any query within a predefined VC-dimension range. Method that collect and store pre-computed statistics gives accurate estimates only for the queries captured by the collected statistics, while estimates of any other query relies on an independence assumption.

To match the accuracy of our new method with histograms and join synopses one would need to create, for each table, a multidimensional histogram where the number of dimensions is equal to the number of columns in the tables. The space needed for a multidimensional histogram is exponential in the number of dimensions, while the size of our sample representation is almost linear in that parameter. Furthermore, to estimate the selectivity for join operations one would need to create join synopses for all pairs of columns in the database, again in space that grows exponential in the number of columns.

It is interesting to note that the highly theoretical concept of VC-dimension leads in this work to an efficient and practical tool for an important data analysis problem.

References

1. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join Synopses for Approximate Query Answering. In: SIGMOD 1999 (1999)
2. Alon, N., Spencer, J.H.: The Probabilistic Method, 3rd edn. John Wiley & Sons, Chichester (2008)
3. Babcock, B., Chaudhuri, S., Das, G.: Dynamic Sample Selection for Approximate Query Processing. In: SIGMOD 2003 (2003)
4. Brown, P.G., Haas, P.J.: Techniques for Warehousing of Sample Data. In: ICDE 2006 (2006)
5. Bruno, N., Chaudhuri, S.: Conditional Selectivity for Statistics on Query Expressions. In: SIGMOD 2004 (2004)
6. Bruno, N., Chaudhuri, S., Gravano, L.: STHoles: a Multidimensional Workload-Aware Histogram. In: SIGMOD 2001 (2001)
7. Chaudhuri, S., Das, G., Narasayya, V.: Optimized Stratified Sampling for Approximate Query Processing. ACM Trans. Database Syst. 32(2), Art. 9 (2007)
8. Chaudhuri, S., Motwani, R., Narasayya, V.: Random Sampling for Histogram Construction: How Much is Enough. In: SIGMOD 1998 (1998)
9. Chaudhuri, S., Motwani, R., Narasayya, V.: On Random Sampling over Joins. In: SIGMOD 1999 (1999)

10. Chazelle, B.: The Discrepancy Method: Randomness and Complexity. Cambridge University Press, Cambridge (2000)
11. Chen, M.C., McNamee, L., Matloff, N.S.: Selectivity Estimation Using Homogeneity Measurement. In: ICDE 1990 (1990)
12. Das, G.: Sampling Methods in Approximate Query Answering Systems. In: Wang, J. (ed.) Encyclopedia of Data Warehousing and Mining. Information Science Publishing (2005)
13. Dobra, A.: Histograms Revisited: When are Histograms the Best Approximation Method for Aggregates over Joins? In: PODS 2005 (2005)
14. Estan, C., Naughton, J.F.: End-Biased Samples for Join Cardinality Estimation. In: ICDE 2006 (2006)
15. Ganguly, S., Gibbons, P.B., Matias, Y., Silberschatz, A.: Bifocal Sampling for Skew-Resistant Join Size Estimation. In: SIGMOD 1996 (1996)
16. Ganti, V., Lee, M.-L., Ramakrishnan, R.: ICICLES: Self-Tuning Samples for Approximate Query Answering. In: VLDB 2000 (2000)
17. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Prentice-Hall, Englewood Cliffs (2002)
18. Gemulla, R., Lehner, W., Haas, P.J.: A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Datasets. In: VLDB 2006 (2006)
19. Getoor, L., Taskar, B., Koller, D.: Selectivity Estimation using Probabilistic Models. In: SIGMOD 2001 (2001)
20. Gibbons, P.B., Matias, Y.: New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In: SIGMOD 1998 (1998)
21. Gryz, J., Liang, D.: Query Selectivity Estimation via Data Mining. In: IIS 2004 (2004)
22. Haas, P.J., Naughton, J.F., Seshadri, S., Swami, A.N.: Fixed-Precision Estimation of Join Selectivity. In: PODS 1993 (1993)
23. Haas, P.J., Naughton, J.F., Swami, A.N.: On the Relative Cost of Sampling for Join Selectivity Estimation. In: PODS 1994 (1994)
24. Haas, P.J., Swami, A.N.: Sequential Sampling Procedures for Query Size Estimation. In: SIGMOD 1992 (1992)
25. Haas, P.J., Swami, A.N.: Sampling-Based Selectivity Estimation for Joins Using Augmented Frequent Value Statistics. In: ICDE 1995 (1995)
26. Haas, P.J., Naughton, J.F., Seshadri, S., Swami, A.N.: Selectivity and Cost Estimation for Joins Based on Random Sampling. Jour. of Comp. and Sys. Sci. 52, 550–569 (1996)
27. Haas, P.J.: Hoeffding Inequalities for Join-Selectivity Estimation and Online Aggregation. IBM Research Report RJ 10040 (2000)
28. Harangsri, B., Shepherd, J., Ngu, A.H.H.: Query Size Estimation using Machine Learning. In: DASFAA 1997 (1997)
29. Har-Peled, S., Sharir, M.: Relative (p, ε)-Approximations in Geometry. Discrete & Computational Geometry 45(3), 462–496 (2011)
30. Hou, W.-C., Ozsoyoglu, G., Dogdu, E.: Error-Constraint Count Query Evaluation in Relational Databases. In: SIGMOD 1991 (1991)
31. Hou, W.-C., Ozsoyoglu, G., Taneja, B.K.: Statistical Estimators for Relational Algebra Expressions. In: PODS 1988 (1988)
32. Ioannidis, Y.E., Poosala, V.: Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In: SIGMOD 1995 (1995)
33. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal Histograms with Quality Guarantees. In: VLDB 1998 (1998)

34. Jin, R., Glimcher, L., Jermaine, C., Agrawal, G.: New Sampling-Based Estimators for OLAP Queries. In: ICDE 2006 (2006)
35. Joshi, S., Jermaine, C.: Robust Stratified Sampling Plans for Low Selectivity Queries. In: ICDE 2008 (2008)
36. Kaushik, R., Naughton, J.F., Ramakrishnan, R., Chakravarthy, V.T.: Synopses for Query Optimization: a Space-Complexity Perspective. ACM Trans. Database Syst. 30(4), 1102–1127 (2005)
37. Larson, P.-A., Lehner, W., Zhou, J., Zabback, P.: Cardinality Estimation Using Sample Views with Quality Assurance. In: SIGMOD 2007 (2007)
38. Li, Y., Long, P.M., Srinivasan, A.: Improved Bounds on the Sample Complexity of Learning. Jour. of Comp. and Sys. Sci. 62, 516–527 (2001)
39. Lipton, R.J., Naughton, J.F.: Query Size Estimation by Adaptive Sampling. J. Comput. Syst. Sci. 51(1), 18–25 (1995)
40. Lipton, R.J., Naughton, J.F., Schneider, D.A.: Practical Selectivity Estimation through Adaptive Sampling. In: SIGMOD 1990 (1990)
41. Löffler, M., Phillips, J.M.: Shape fitting on point sets with probability distributions. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 313–324. Springer, Heidelberg (2009)
42. Markl, V., Haas, M., Peter, J., Kutsch, Megiddo, N., Srivastava, U., Tran, T.M.: Consistent Selectivity Estimation via Maximum Entropy. The VLDB Journal 16(1), 55–76 (2007)
43. Matias, Y., Vitter, J.S., Wang, M.: Wavelet-Based Histograms for Selectivity Estimation. In: SIGMOD 1998 (1998)
44. Matoušek, J.: Lectures on Discrete Geometry. Springer, Heidelberg (2002)
45. Miller, R.J.: Simultaneous Statistical Inference, 2nd edn. Springer, Heidelberg (1981)
46. Ngu, A.H., Harangsri, B., Shepherd, J.: Query Size Estimation for Joins Using Systematic Sampling. Distributed and Parallel Databases 15, 237–275 (2004)
47. Olken, F.: Random Sampling from Databases. Ph.D. dissertation, LBL Tech. Report LBL-32883 (1993)
48. Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: Improved Histograms for Selectivity Estimation of Range Predicates. In: SIGMOD 1996 (1996)
49. Poosala, V., Ioannidis, Y.E.: Selectivity Estimation without the Attribute Value Independence Assumption. In: VLDB 1997 (1997)
50. Ré, C., Suciu, D.: Understanding Cardinality Estimation Using Entropy Maximization. In: PODS 2010 (2010)
51. Riondato, M., Akdere, M., Çetintemel, U., Zdonik, S.B., Upfal, E.: The VC-Dimension of SQL Queries and Selectivity Estimation Through Sampling. CoRR abs/1101.5805 (2011)
52. Srivastava, U., Haas, P.J., Markl, V., Kutsch, M., Tran, T.: ISOMER: Consistent Histogram Construction Using Query Feedback. In: ICDE 2006 (2006)
53. Vapnik, V.N., Chervonenkis, A.Y.: On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. Theory of Probability and its Applications 16(2), 264–280 (1971)
54. Wang, H., Sevcik, K.C.: A Multi-Dimensional Histogram for Selectivity Estimation and Fast Approximate Query Answering. In: CASCON 2003 (2003)
55. Wang, M., Vitter, J.S., Iyer, B.R.: Selectivity Estimation in the Presence of Alphanumeric Correlations.In: ICDE 1997 (1997)
56. Wu, Y.-L., Agrawal, D., El Abbadi, A.: Applying the Golden Rule of Sampling for Query Estimation. In: SIGMOD 2001 (2001)