Journal of Machine Learning Research X (201X) X-XX Submitted 10/13; Published X/XX

The VC-Dimension of SQL Queries and Selectivity Estimation Through Sampling

Matteo Riondato MATTEO@CS.BROWN.EDU
Mert Akdere MAKDERE@CS.BROWN.EDU
Uğur Çetintemel UGUR@CS.BROWN.EDU
Stanley B. Zdonik SBZ@CS.BROWN.EDU
Eli Upfal ELI@CS.BROWN.EDU
Department of Computer Science
Brown University
Providence, RI, USA

Editor:

Abstract

In this work we show how *Vapnik-Chervonenkis (VC) dimension*, a fundamental result in statistical learning theory, can be used to evaluate the selectivity (output cardinality) of SQL queries, a core problem in large database management.

The major theoretical contribution of this work, which is of independent interest, is an explicit bound to the VC-dimension of a range space defined by all possible outcomes of a collection of queries. We prove that the VC-dimension can be bounded by a quantity that 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 (number of tuples) of the database.

Leveraging on this result we develop a method that, given a class of queries, builds a concise random sample of a database that is small enough to be stored in main memory and is 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.

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 Microsoft SQL Server.

Keywords: VC-dimension, SQL queries, databases, sampling, selectivity estimation.

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. The task of efficiently obtaining such accurate estimates has been extensively studied in previous work with solutions ranging from storage of pre-computed statistics on the distribution of values in the tables, to online sampling of the databases, and to combinations of the two approaches (Lipton and Naughton, 1995; Lipton et al., 1990; Haas and Swami, 1992; Hou et al., 1991; Haas and Swami, 1995; Ganguly et al., 1996; Ganti et al., 2000; Gibbons and Matias, 1998; Hou et al., 1988; Larson et al., 2007; Poosala and Ioannidis, 1997). Histograms, simple yet powerful statistics of the data in the tables, are the most commonly used solution in practice, thanks 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 then performing the analysis itself. This approach is therefore usually more expensive than a histogram lookup. Our goal in this work is to exploit both the computational efficiency of using pre-collected data and the provable accuracy of estimates obtained by running a query on a properly sized random sample of the database.

We apply the statistical concept of Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis, 1971) to develop and analyze a novel technique to generate 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. 3.2 for formal definitions). A major theoretical contribution of this work, which is of independent interest, is an explicit bound to the VC-dimension of various classes of queries, viewed as indicator functions on the Cartesian product of the database tables. In particular, we show an upper bound to the VC-dimension of a class of queries that 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 family of queries, defined by its VC-dimension, builds a concise sample of the 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 can be obtained using a sample of a surprisingly small size (see Table 2 for concrete values), which can then reside in main memory, with the net result of a significant speedup in the execution of queries on the sample.

A technical difficulty in applying the VC-dimension results to the database setting is that they assume the availability of a uniform sample of the Cartesian product of all the tables, while in practice it is more efficient to store a sample of each table separately and run the queries on the Cartesian product of the samples, which has a different distribution

than a sample of the Cartesian product of the tables. We develop an efficient procedure for constructing a sample that circumvents this problem (see Sect. 5).

We present extensive experimental results that validate our theoretical analysis and demonstrate 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 method is that it gives provably accurate predictions for the selectivities of all queries with up to a given complexity (VC-dimension) specified by the user before creating the sample, 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. Das (2009) presents a survey of the possible solutions to this latter task.

Outline. The rest of the paper is organized as follows. We review the relevant previous work in Sect. 2. In Sect. 3 we formulate the problem and introduce the Vapnik-Chervonenkis dimension and the related tools we use in developing our results. Our main analytical contribution, a bound on the VC dimension of class of queries is presented in Sect. 4. The application of these results for selectivity estimation is given in Sect. 5. Experiments are presented in Sect. 6.

2. Related Work

Methods to estimate the selectivity (or cardinality of the output) of queries have been extensively studied in the database literature primarily due to the importance of this task to query plan optimization and resource allocation. A variety of approaches have been explored, ranging from the use of sampling, both online and offline, to the pre-computation of different statistics such as histograms, to the application of methods from machine learning (Chen et al., 1990; Harangsri et al., 1997), data mining (Gryz and Liang, 2004), optimization (Chaudhuri et al., 2007; Markl et al., 2007), and probabilistic modeling (Getoor et al., 2001; Ré and Suciu, 2010).

The use of sampling for selectivity estimation has been studied mainly in the context of online sampling (Lipton et al., 1990; Lipton and Naughton, 1995), where a sample is obtained, one tuple at a time, after the arrival of a query and it used only to evaluate the selectivity of that query and then discarded. Sampling at random from a large database residing on disk is an expensive operation (Olken, 1993; Brown and Haas, 2006; Gemulla et al., 2006), and in some cases sampling for an accurate cardinality estimate is not significantly faster than full execution of the query (Haas et al., 1993, 1994).

A variety of sampling and statistical analysis techniques has been tested to improve the efficiency of the sampling procedures and in particular to identify early stopping conditions. These include sequential sampling analysis (Hou et al., 1991; Haas and Swami, 1992), keeping additional statistics to improve the estimation (Haas and Swami, 1995), labelling the tuples and using label-dependent estimation procedures (Ganguly et al., 1996), or applying the cumulative distribution function inversion procedure (Wu et al., 2001). Some works also looked at non-uniform sampling (Babcock et al., 2003; Estan and Naughton, 2006) and stratified sampling (Chaudhuri et al., 2007; Joshi and Jermaine, 2008). Despite all these relevant contributions, online sampling is still considered too expensive for most applications.

An off-line sampling approach was explored by Ngu et al. (2004), 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. Their work does not give any explicit guarantee on the accuracy of the predictions. Chaudhuri et al. (2007) 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 maximum error. Haas (1996) 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 (Miller, 1981). In contrast, our result holds 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. (1999), the Cartesian product of uniform samples of a number of tables is different from a uniform sample of the Cartesian product of those tables. Furthermore, given a size s, it is impossible to a priori determine two sample sizes s_1 and s_2 such that uniform 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 (Hou et al., 1988; Gibbons and Matias, 1998; Ganti et al., 2000; Jin et al., 2006; Larson et al., 2007), with histograms being the most commonly used representation. The construction, maintenance, and use of histograms were thoroughly examined in the literature (Jagadish et al., 1998; Ioannidis and Poosala, 1995; Matias et al., 1998; Poosala et al., 1996), with both theoretical and experimental results. In particular Chaudhuri et al. (1998) 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. (2005) 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 because of two major drawbacks in the use of histograms: 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 (Bruno and Chaudhuri, 2004; Dobra, 2005; Poosala and Ioannidis, 1997; Wang et al., 1997; Wang and Sevcik, 2003). Among these solutions, the use of multidimensional histograms (Bruno et al., 2001; Poosala and Ioannidis, 1997; Srivastava et al., 2006; Wang and Sevcik, 2003) seems the most practical. Nevertheless, these techniques are not widespread due to the extra memory and computational costs in their implementation.

Efficient and practical techniques for drawing random samples from a database and for updating the sample when the underlying tables evolve have been extensively analyzed in the database literature (Brown and Haas, 2006; Gemulla et al., 2006, 2007; Haas and König, 2004; Jermaine et al., 2004).

The *Vapnik-Chervonenkis dimension* was first introduced in a seminal article (Vapnik and Chervonenkis, 1971) on the convergence of sample averages to their expectations, but it was only with the work of Haussler and Welzl (1986) and Blumer et al. (1989) that it was applied to learning. Since then, VC-dimension has encountered enormous success and application in the fields of computational geometry (Chazelle, 2000; Matoušek, 2002) and machine learning (Anthony and Bartlett, 1999; Mohri et al., 2012) but its use in system-related fields is not as widespread. In the database literature, it was used in the context of constraint databases to compute good approximations of aggregate operators (Benedikt and Libkin, 2002). VC-dimension-related results were also recently applied in the field of database privacy by Blum et al. (2008) to show a bound on the number of queries needed for an attacker to learn a private concept in a database. Gross-Amblard (2011) showed that content with unbounded VC-dimension can not be watermarked for privacy purposes. Riondato and Upfal (2012) used VC-dimension to compute a sample size sufficient for computing high-quality approximations of the collections of Frequent Itemsets and Association Rules in a transactional dataset.

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

3. Preliminaries

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

3.1 Database Queries and Selectivity

We outline here some basic definitions about databases, queries, and selectivity. We refer the reader to complete textbooks for additional information (Garcia-Molina et al., 2002). We consider a *database* $\mathcal{D}$ of k tables $\mathcal{T}_1, \ldots, \mathcal{T}_k$. A *table* is a two-dimensional representation of data. Its rows are called *tuples* and it can 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 $D(\mathcal{T}.C)$. Table 1 shows two examples of database tables, `Customers` and `CarColors`. 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. In full generality, $\mathcal{C}$ is the Boolean combination of clauses of the form $\mathcal{T}.C_i \,\mathsf{op}\, a_i$, 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 $\mathcal{T}.C_i$.*

Customers

Name	*Street*	*ZipCode*	*PhoneNumber*
John Doe	Pratt Av.	02906	401-1234567
Jim Clark	Morris Rd.	02906	502-8902134
Greta Garbo	Pitman St.	05902	853-9876543

CarColor

Name	*Color*
John Doe	Blue
Jane Doe	Red
Greta Garbo	Red

Table 1: Example of database tables.

As an example, the selection query

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \texttt{WHERE } \textit{Customers.ZipCode} = 02906$$

would return the first and second row of Table 1.

We assume that all $D(\mathcal{T}.C_i)$ are such that it is possible to build total order relations on them. 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*

$$\{(t_1, t_2) \ : \ t_1 \in \mathcal{T}_1, t_2 \in \mathcal{T}_2, \ \textit{s.t.} \ t_1.C \, \mathsf{op} \, t_2.C\}$$

where "op*" is one of* $\{<, >, \geq, \leq, =, \neq\}$.

An example of a join query is the following:

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \textit{CarColors } \texttt{WHERE } \textit{Customers.Name} = \textit{CarColors.Name}$$

This query would return the following tuples:

$$(\text{John Doe}, \text{Pratt Av.}, 02906, \text{401-1234567}, \text{Blue}),$$
$$(\text{Greta Garbo}, \text{Pitman St.}, 05902, \text{853-9876543}, \text{Red})$$

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

Our definition of a join query is basically equivalent to that of a *theta-join* (Garcia-Molina et al., 2002, 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 $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$, 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 $S_1, \ldots, S_\ell$.*

As an example, the query

$$\texttt{SELECT} * \texttt{FROM } \textit{Customers}, \textit{CarColors} \texttt{ WHERE } \textit{Customers.Name} = \textit{CarColors.Name}$$
$$\texttt{AND } \textit{Customers.ZipCode} = 02906 \texttt{ AND } \textit{CarColors.Color} = \textit{Red}$$

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.

Definition 4 *Given a query q, a* query plan *for q is a directed binary tree T_q whose nodes are the elementary (i.e., select or join) operations into which q can be decomposed. There is an edge from a node a to a node b if the output of a is used as an input to b. The operations on the leaves of the tree use one or two tables of the database as input. The output of the operation in the root node of the tree is the output of the query.*

It follows from the definition of a combination of select and join operations that a query may conform with multiple query plans. Nevertheless, for all the queries we defined there is (at least) one query plan such that all select operations are in the leaves and internal nodes are join nodes (Garcia-Molina et al., 2002). To derive our results, we use these specific query 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 query, pairs of tuples if q is a join query, and ℓ-uples of tuples for combinations of join and select involving ℓ tables) 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.*

Our goal is to store a succinct representation (sample) $\mathcal{S}$ of the database $\mathcal{D}$ such that an execution of a query on the sample $\mathcal{S}$ will provide an accurate estimate for the selectivity of each operation in the query plan when executed on the database $\mathcal{D}$.

3.2 Vapnik-Chervonenkis Dimension

The Vapnik-Chernovenkis (VC) Dimension of a family F of indicator functions (or equivalently a family of subsets) on a domain is a measure of the complexity or expressiveness of F (Vapnik and Chervonenkis, 1971). A finite bound on the VC-dimension of F implies a bound on the number of random samples required for approximately learning a function from F. We outline here 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 (2008, Sect. 14.4), Chazelle (2000, Chap. 4), Mohri et al. (2012), and Vapnik (1999) for introductions and 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 (as sets of tuples) of the queries in Q when run 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.

To define the VC-dimension of a range space we consider the projection of the ranges into a set of points:

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

A set is said to be shattered if all its subsets are defined by the range space:

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.*

The VC-dimension of a range space is the cardinality of the largest set shattered by the space:

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

An example is shown in Fig. 1.

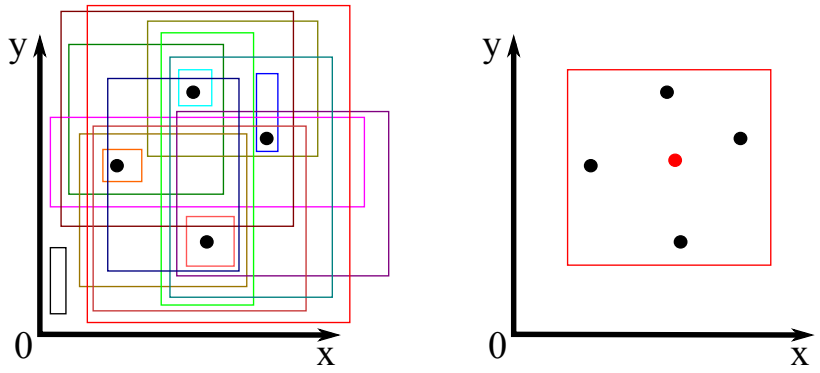

Figure 1: Example of range space and VC-dimension. The space of points is the plane $\mathbb{R}^2$ and the set of ranges is the set of all *axis-aligned rectangles*. The figure on the left shows graphically that it is possible to shatter a set of four points using 16 rectangles. On the right instead, 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 $\mathsf{VC}((X, R)) = 4$.

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.

Note that a range space (X, R) with an arbitrarily large set of points X and an arbitrarily 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 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 < 3. This observation is generalized in the following result (Matoušek, 2002, Lemma 10.3.1):

Lemma 10 *The VC-Dimension of the range space $(\mathbb{R}^d, X)$, where X is the set of all half-spaces in $\mathbb{R}^d$ equals $d + 1$.*

The main application of VC-dimension in statistics and learning theory is its relation to the minimum size sample needed for approximate learning of a set of indicator functions or hypotheses.

Definition 11 *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 in (X, R) if $\forall r \in R$, 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 in (X, R) 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 $\frac{|B \cap r|}{|B|} \leq (1 + \varepsilon)p$.*

An ε-approximation (resp. a relative (p, ε)-approximation) can be probabilistically constructed by sampling the point space (Vapnik and Chervonenkis, 1971; Li et al., 2001; Har-Peled and Sharir, 2011).

Theorem 12 *Let (X, R) be a range space of VC-dimension at most d, A a finite subset of X, and $B \subset A$ a random sample of A of cardinality s.*

1. *There is a positive constant c such that for any $0 < \varepsilon, \delta < 1$ and*

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

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

2. *There is a positive constant c' such that for any $0 < p < 1$ and*

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

   B is a relative (p, ε)-approximation for A with probability at least $1 - \delta$.

Löffler and Phillips (Löffler and Phillips, 2009) estimated experimentally that the constant c is at most 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 $O(d^{3d}(\frac{1}{\varepsilon^2}\log\frac{d}{\varepsilon})^d|X|)$ (Chazelle, 2000).

In Sect. 5 we use an ε-approximation (or a relative (p,ε)-approximation) to compute good estimates of the selectivities of all queries in Q. We obtain a small approximation set through probabilistic construction. The challenge in applying Thm. 12 to our setting is computing the VC-dimension of a range space defined by a class of queries. We state here a few fundamental results that will be used in our analysis.

First, it is clear that if the VC-dimension of a range space (X,R) is d then $2^d \leq |R|$ since all subsets of some set of size d are defined by members of R. Next we define for integers $n > 0$ and $d > 0$ the *growth function* $g(d,n)$ as

$$g(d,n) = \sum_{i=0}^{d} \binom{n}{i} < n^d.$$

The growth function is used in the following results (Alon and Spencer, 2008, Sect. 14.4).

Lemma 13 (Sauer's Lemma) *If (X,R) is a range space of VC-dimension d with $|X| = n$ points, then $|R| \leq g(d,n)$.*

Corollary 14 *If (X,R) is a range space of VC-dimension d, then for every finite $A \subset X$, $|P_R(A)| \leq g(d,|A|)$.*

Our analysis in Sect. 4 uses the following bound which is an extension of (Alon and Spencer, 2008, Corol. 14.4.3) to arbitrarily combinations of set operations.

Lemma 15 *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 includes all possible combinations of union and intersections of h members of R. Then $\mathsf{VC}(X,R_h) \leq 3dh\log(dh)$.*

Proof Let A be an arbitrarily subset of cardinality n of X. We have $|P_R(A)| \leq g(d,n) \leq n^d$. There are

$$\binom{|P_R(A)|}{h} \leq \binom{g(d,n)}{h} \leq n^{dh}$$

possible choices of h members of $P_R(A)$, and there are no more than

$$h!2^{h-1}C_{h-1} \leq h^{2h}$$

Boolean combinations using unions and intersections of the h sets, where C_{h-1} is the $(h-1)^{\text{th}}$ Catalan number ($C_i = \frac{1}{i+1}\binom{2i}{i}$). If

$$2^n > h^{2h}n^{dh} \geq |P_{R_h}(A)|$$

then A cannot be shattered. This inequality holds for $n \geq 3dh\log(dh)$. To see this, consider the fact that clearly $2^n \geq h^{2h}n^{dh}$ for sufficiently large n, so it suffices to show that $n = 3dh\log(dh)$ is sufficiently large. Next observe that the function $f(x) = 2\log x - \log(3x\log x)$ is positive for $x \geq 2$ since it is positive for $x = 2$ and it is easy to see that the derivative $f'(x)$

is positive for $x \geq 2$. It immediately follows that $3\log(dh) > \log(dh) + \log(3dh\log(dh)$ for $d \geq 2$ and $h \geq 1$. Singe $dh\log(dh) \geq 2h\log(h)$, we have $3dh\log(dh) > 2h\log(3dh\log(dh))$, which proves the result. ■

4. The VC-dimension of Classes of Queries

In this section we develop a general bound to the VC-dimension of classes of queries. We start by computing the VC-dimension of simple select queries on one column and then move to more complex queries (multi-attributes select queries, join queries). We then extend our bounds to general queries that are combinations of multiple select and join operations.

4.1 Select Queries

Let $\mathcal{T}$ be a table with m columns $\mathcal{T}.C_1, \ldots, \mathcal{T}.C_m$, and n tuples. For a fixed column $\mathcal{T}.C$, consider the set Σ_C of the selection queries in the form

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

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

Let $q_1, q_2 \in \Sigma_C$ be two queries. We say that q_1 is equivalent to q_2 (and denote this fact as $q_1 = q_2$) if their outputs are identical, i.e., they return the same set of tuples when they are run on the same database. Note that $q_1 = q_2$ defines a proper equivalence relation.

Let $\Sigma^*_C \subseteq \Sigma_C$ be a maximum subset of Σ_C that contains no equivalent queries, i.e., it contains one query from each equivalent class.

Lemma 16 *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 $m + 1$.

Proof We can view the tuples of $\mathcal{T}$ as points in the m-dimensional space $\Delta = D(\mathcal{T}.C_1) \times D(\mathcal{T}.C_2) \times \ldots \times D(\mathcal{T}.C_m)$. A tuple $t \in \mathcal{T}$ such that $t.C_1 = a_1, t.C_2 = a_2, \ldots, t.C_m = a_m$ is represented on the space by the point $(a_1, a_2, \ldots, a_m)$.

The queries in $\Sigma^*_{\mathcal{T}}$ can be seen as half spaces of Δ. In particular any query in $\Sigma^*_{\mathcal{T}}$ is defined as in (2) and can be seen as the half space

$$\{(x_1, \ldots, x_i, \ldots, x_m) \ : \ x_j \in D(\mathcal{T}_j) \text{ for } j \neq i, \text{ and } x_i \, \mathsf{op} \, a_i\} \subseteq \Delta \ .$$

It then follows from Lemma 10 that $\mathsf{VC}(S) \leq m + 1$. ■

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

We now extend these result to general selection queries. Consider the set $\Sigma_{\mathcal{T}}^{2*}$ of queries whose selection predicate can be expressed as the Boolean combination of the selection predicates of at most two queries from $\Sigma_{\mathcal{T}}^{*}$. These are the queries of the form:

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T} \ \texttt{WHERE} \ \mathcal{T}.X_1 \, \mathsf{op}_1 \, a_1 \, \mathsf{bool} \, \mathcal{T}.X_2 \, \mathsf{op}_2 \, a_2$$

where $\mathcal{T}.X_1$ and $\mathcal{T}.X_2$ are two columns from $\mathcal{T}$ (potentially, $\mathcal{T}.X_1 = \mathcal{T}.X_2$), $a_1 \in D(\mathcal{T}.X_1)$, $a_2 \in D(\mathcal{T}.X_2)$, "$\mathsf{op}_i$" is either "$\geq$" or "$\leq$" and "$\mathsf{bool}$" is either "`AND`" or "`OR`". Note that, in particular the queries in the form

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T} \ \texttt{WHERE} \ \mathcal{T}.X_1 \, \mathsf{eqop} \, a$$

where eqop is either "$=$" or "$\neq$", belong to $\Sigma_{\mathcal{T}}^{2*}$ because we can rewrite a selection predicate containing one of these operators as a selection predicate of two clauses using "$\geq$" and "$\leq$" joined by either AND (in the case of "$=$") or OR (in the case of "$\neq$").

By applying Lemma 15, we have that the VC-dimension of the range space $(\mathcal{T}, \Sigma_{\mathcal{T}}^{2*})$ is at most $3(m+1)2\log((m+1)2)$, where m is the number of columns in the table $\mathcal{T}$.

We can generalize this result to b Boolean combinations of selection predicates as follows.

Lemma 17 *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 $3((m+1)b)\log((m+1)b)$.*

Note that we can not apply the bound used in the proof of Lemma 16 since not all queries in $\Sigma_{\mathcal{T}}^{b*}$ are equivalent to axis-aligned boxes. Once we apply Boolean operations on the outputs of the individual select operation, the set of possible outputs, $S_b = (\mathcal{T}, \Sigma_{\mathcal{T}}^{b*})$, may form complex subsets, including unions of disjoint (half-open) axis aligned-rectangles and/or intersections of overlapping ones that cannot be represented as a collection of half spaces. Thus, we need to apply a different technique here.

Proof The output of a query q in $\Sigma_{\mathcal{T}}^{b^*}$ can be seen as the Boolean combination (i.e., union and intersection) of the outputs of at most b "simple" select queries q_i from $\Sigma_{\mathcal{T}}^{*}$ where each of these queries q_i is as in (2). An `AND` operation in the predicate of q implies an intersection of the outputs of the corresponding two queries q_i and q_j, while an `OR` operation implies a union of the outputs. The thesis follows by applying Lemma 15. ■

4.2 Join Queries

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two distinct tables, and let R_1 and R_2 be two families of (outputs of) select 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 $\mathsf{VC}(S_1), \mathsf{VC}(S_2) \geq 2$. 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

$$J_{r_1,r_2}^{\mathsf{op}} = \{(t_1, t_2) \ : \ t_1 \in r_1, t_2 \in r_2, t_1.C \, \mathsf{op} \, t_2.C\},$$

where $\mathsf{op} \in \{>, <, \geq, \leq, =, \neq\}$. $J^{\mathsf{op}}_{r_1,r_2}$ is the set of ordered pairs of tuples (one from $\mathcal{T}_1$ and one from $\mathcal{T}_2$ that forms the output of the join query

$$\texttt{SELECT} \ * \ \texttt{FROM} \ \mathcal{T}_1, \mathcal{T}_2 \ \texttt{WHERE} \ r_1 \ \texttt{AND} \ r_2 \ \texttt{AND} \ \mathcal{T}_1.C \, \mathsf{op} \, \mathcal{T}_2.C \ . \tag{3}$$

Here we simplify the notation by identifying select queries with their predicates. We have $J^{\mathsf{op}}_{r_1,r_2} \subseteq r_1 \times r_2$ and $J^{\mathsf{op}}_{r_1,r_2} \subseteq T_J$. Let

$$J_C = \{J^{\mathsf{op}}_{r_1,r_2} \mid r_1 \in R_1, r_2 \in R_2, \mathsf{op} \in \{>, <, \geq, \leq, =, \neq\}\}.$$

J_C is the set of outputs of all join queries like the one in (3), for all pairs of queries in $R_1 \times R_2$ and all values of "op". We present here an upper bound to the VC-dimension of the range space $S_J = (T_J, J_C)$.

Lemma 18 $\mathsf{VC}(S_J) \leq 3(\mathsf{VC}(S_1) + \mathsf{VC}(S_2)) \log((\mathsf{VC}(S_1) + \mathsf{VC}(S_2)))$.

Proof Let $v_1 = \mathsf{VC}(S_1)$ and $v_2 = \mathsf{VC}(S_2)$. Assume that a set $A \subseteq T_J$ is shattered by J_C, and $|A| = v$. Consider the two cross-sections $A_1 = \{x \in \mathcal{T}_1 \ : \ (x, y) \in A\}$ and $A_2 = \{y \in \mathcal{T}_2 \ : \ (x, y) \in A\}$. Note that $|A_1| \leq v$ and $|A_2| \leq v$ and by 14 $|P_{R_1}(A_1)| \leq g(v_1, v) \leq v^{v_1}$ and $|P_{R_2}(A_2)| \leq g(v_2, v) \leq v^{v_2}$. For each set $r \in P_{J_C}(A)$ (i.e., for each subset $r \subseteq A$, given that $P_{J_C}(A) = 2^A$) there is a pair (r_1, r_2), $r_1 \in R_1$, $r_2 \in R_2$, and there is op, such that $r = A \cap J^{\mathsf{op}}_{r_1,r_2}$. Each of such pair (r_1, r_2) identifies a distinct pair $(r_1 \cap A_1, r_2 \cap A_2) \in P_{R_1}(A_1) \times P_{R_2}(A_2)$, therefore each element of $P_{R_1}(A_1) \times P_{R_2}(A_2)$ can be identified at most 6 times, the number of possible values for "op".

In particular, for a fixed "op", an element of $P_{R_1}(A_1) \times P_{R_2}(A_2)$ can be identified at most once. To see this, consider two different sets $s_1, s_2 \in P_{J_C}(A)$. Let the pairs (a_1, a_2), (b_1, b_2), $a_1, b_1 \in R_1$, $a_2, b_2 \in R_2$, be such that $s_1 = A \cap J^{\mathsf{op}}_{a_1,a_2}$ and $s_2 = A \cap J^{\mathsf{op}}_{b_1,b_2}$. Suppose that $a_1 \cap A_1 = b_1 \cap A_1$ $(\in P_{R_1}(A_1))$ and $a_2 \cap A_2 = b_2 \cap A_2$ $(\in P_{R_2}(A_2))$. The set s_1 can be seen as $\{(t_1, t_2) \ : \ t_1 \in a_1 \cap A_1, t_2 \in a_2 \cap A_2 \text{ s.t. } t_1.C \, \mathsf{op} \, t_2.C\}$. Analogously the set s_2 can be seen as $\{(t_1, t_2) \ : \ t_1 \in b_1 \cap A_1, t_2 \in b_2 \cap A_2 \text{ s.t. } t_1.C \, \mathsf{op} \, t_2.C\}$. But given that $a_1 \cap A_1 = b_1 \cap A_1$ and $a_2 \cap A_2 = b_2 \cap A_2$, this leads to $s_1 = s_2$, a contradiction. Hence, a pair (c_1, c_2), $c_1 \in P_{R_1}(A_1)$, $c_2 \in P_{R_2}(A_2)$ can only be identified at most 6 times, one for each possible value of "op".

Thus,$|P_{J_C}(A)| \leq 6|P_{R_1}(A_1)| \cdot |P_{R_2}(A_2)|$. A could not be shattered if $|P_{J_C}(A)| < 2^v$. Since

$$|P_{J_C}(A)| \leq 6|P_{R_1}(A_1)| \cdot |P_{R_2}(A_2)| \leq 6g(v_1, v)g(v_2, v) \leq 6v^{v_1+v_2},$$

it is sufficient to have $6v^{v_1+v_2} \leq 2^v$, which holds for $v > 3(v_1 + v_2) \log(v_1 + v_2)$. ■

The above results can be generalized to any query plan represented as a tree where the select operations are in the leaves and all internal nodes are join operations. As we said earlier, such a tree exists for any query.

Lemma 19 *Consider the class Q of queries that can be seen as combinations of select and joins 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 $v_i = \mathsf{VC}(S_i)$. 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*

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

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

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

Note that this Lemma is not just an extension of Lemma 18 to join queries between two multicolumns tables. Instead, it is an extension to queries containing joins between multiple tables (possibly between multicolumns tables).

Proof Assume that a set $A \subseteq T_J$ is shattered by R_Q, and $|A| = v$. Consider the cross-sections $A_i = \{x \in \mathcal{T}_i \ : \ (y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, y_u) \in A\}, 1 \leq i \leq u$. Note that $|A_i| \leq v$ and by 14 $|P_{R_i}(A_i)| \leq g(v_i, v) \leq v^{v_i}$. For each set $r \in P_{J_C}(A)$ (i.e., for each subset $r \subseteq A$, given that $P_{J_C}(A) = 2^A$) there is a sequence $\rho = (r_1, \ldots, r_u)$, $r_i \in R_i$, and there is an ω, such that $r = A \cap J_\rho^\omega$. Each sequence ρ identifies a distinct sequence $(r_1 \cap A_1, r_2 \cap A_2, \ldots, r_u \cap A_u) \in P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$, therefore each element of $P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$ can be identified at most 6^{u-1} times, one for each different ω.

In particular, for a fixed ω, an element of $P_{R_1}(A_1) \times \cdots \times P_{R_u}(A_u)$ can be identified at most once. To see this, consider two different sets $s_1, s_2 \in P_{J_C}(A)$. Let the vectors $\rho_a = (a_1, \ldots, a_u)$, $\rho_b = (b_1, \ldots, b_u)$, $a_i, b_i \in R_i$, be such that $s_1 = A \cap J_{\rho_a}^\omega$ and $s_2 = A \cap J_{\rho_b}^\omega$. Suppose that $a_i \cap A_i = b_i \cap A_i$ $(\in P_{R_i}(A_i))$. The set s_1 can be seen as $\{(t_1, \ldots, t_u) \ : \ t_i \in a_i \cap A_i, \ \text{s.t.} \ (t_1, \ldots, t_u) \ \text{satisfies} \ \omega\}$. Analogously the set s_2 can be seen as $\{(t_1, \ldots, t_u) \ : \ t_i \in b_i \cap A_i, \ \text{s.t.} \ (t_1, \ldots, t_u) \ \text{satisfies} \ \omega\}$. But given that $a_i \cap A_i = b_i \cap A_i$, this leads to $s_1 = s_2$, a contradiction. Hence, a vector $(c_1, \ldots, c_u)$, $c_i \in P_{R_i}(A_i)$, can only be identified at most a finite number ℓ of times, once for each different ω. For each of the $u-1$ join conditions composing ω we need to choose a pair $(\mathcal{T}_1.A, \mathcal{T}_1.B)$ expressing the columns along which the tuples should be joined. There are at most $g = \binom{um}{2}$ such pairs (some of them cannot actually be chosen, e.g., those of the type $(\mathcal{T}_1.A, \mathcal{T}_1.B)$). There are then $\binom{g}{u-1}$ ways of choosing these $u-1$ pairs. For each choice of $u-1$ pairs, there are $6^{(u-1)}$ ways of choosing the operators in the join conditions (6 choices for op for each pair). We have

$$\ell \leq \binom{\binom{um}{2}}{u-1} \cdot 6^{(u-1)} \leq (mu)^{2u}.$$

Thus, $|P_{J_C}(A)| \leq \ell |P_{R_1}(A_1)| \cdot \cdots \cdot |P_{R_u}(A_u)|$. A could not be shattered if $|P_{J_C}(A)| < 2^v$. Since we have

$$|P_{J_C}(A)| \leq \ell \cdot |P_{R_1}(A_1)| \cdot \cdots \cdot |P_{R_u}(A_u)| \leq \ell \cdot g(v_1, v) g(v_2, v) \ldots g(v_u, v) \leq$$
$$\leq (mu)^{2u} \cdot v^{v_1 + \cdots + v_u},$$

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

then it is sufficient to have

$$(mu)^{2u} v^{\sum_{i=1}^{v} vi} \leq 2^v,$$

which holds for $v > 4u\,(\sum_i v_i)\log(u\sum_i v_i)$. ■

4.3 General Queries

Combining the above results we prove:

Theorem 20 *Consider a 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*

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

Note that Thm. 20 gives an upper bound to the VC-dimension. Our experiments suggest that in most cases the VC-dimension and the corresponding minimum sample size are even smaller.

5. Estimating Query Selectivity

We apply the theoretical result on the VC-dimension of queries to constructing a concrete algorithm for selectivity estimation and query plan optimization.

5.1 The general scheme

Our goal is to apply Def. 11 and Thm. 12 to compute an estimate of the selectivity of SQL queries. Let $Q_{u,m,b}$ be a class of queries as in Thm. 20. The class $Q_{u,m,b}$ defines a range space $S = (X, R)$ such that X is the Cartesian product of the tables involved in executing queries in $Q_{u,m,b}$, and R is the family of all output sets of queries in $Q_{u,m,b}$. Let $\mathcal{S}$ be an ε-approximation of X and let r be the output set of a query $q \in Q_{u,m,b}$ when executed on the original dataset, then $X \cap r = r$ and $r \cap \mathcal{S}$ is the output set when the query is executed on the sample (see details below). Thus, by Def. 11,

$$\left|\frac{|X \cap r|}{|X|} - \frac{|\mathcal{S} \cap r|}{|\mathcal{S}|}\right| = |\sigma_{\mathcal{D}}(q) - \sigma_{\mathcal{S}}(q)| \leq \varepsilon,$$

i.e., the selectivity of running a query $q \in Q_{u,m,b}$ on an ε-approximation of X is within ε of the selectivity of q when executed on the original set of tables. 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 an ε-approximation of X we obtain accurate estimates for the selectivity of all the subqueries defined by its execution plan. Corresponding results are obtained by using a relative (p, ε)-approximation for X.

5.2 Building and using the sample representation

We apply Thm. 12 to probabilistically construct an ε-approximation of X. A technical difficulty in algorithmic application of the theorem is that it is proven for a uniform sample of the Cartesian product of all the tables in the database, while 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 (Chaudhuri et al., 1999). We developed the following procedure to circumvent this problem. 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 (with replacement) 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 $x_1.sampleindex = x_2.sampleindex = \cdots = x_\ell.sampleindex = i$. 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 (see Algorithms 1 and 2).

Algorithm 1: `CreateSample`$(s, (T_1, \ldots, T_k))$

```
input  : sample size s, tables T_1, ..., T_k.
output: sample tables S_1, ..., S_k with t tuples each.
for j ← 1 to k do
    S_j ← ∅
end
for i ← 1 to s do
    for j ← 1 to k do
        t ← drawRandomTuple(T_j)
        t.sampleindex_j ← i
        S_j ← S_j ∪ {t}
    end
end
```

Theorem 21 *The* `ComputeSelectivity` *procedure (in Alg. 2) executes a query on the Cartesian product of independent random samples of the tables but outputs the selectivity that corresponds to executing the query on a random sample of the Cartesian product of the original tables.*

Proof The *CreateSample* procedure chooses from each table a random sample of t tuples and adds to each sampled tuple an index in $[1, t]$. Each sample table has exactly one tuple

Algorithm 2: `ComputeSelectivity`$(\mathcal{S}, op)$

input : elementar database operation op, sample database $S = (\mathcal{S}_1, \ldots, \mathcal{S}_k)$ of size s.
output: the selectivity of op.
$O_{op} \leftarrow$ `executeOperation(`S, op`)`;
$(\ell_1, \ldots, \ell_j) \leftarrow$ indexes of the sample tables involved in op ;
$i \leftarrow 0$;
for $tuple \in O_{op}$ **do**
  if $tuple.sampleindex_{\ell_1} = tuple.sampleindex_{\ell_2} = \cdots = tuple.sampleindex_{\ell_j}$ **then**
  $i \leftarrow i + 1$;
end
$selectivity \leftarrow i/s$;

with each index value, and the Cartesian product of the sample tables has exactly one element that is a concatenation of tuples, all with the same index i in their tables. Restricting the selectivity computation to these t elements (as in *ComputeSelectivity*) gives the result. ■

Note that our method circumvent the major difficulty pointed out by Chaudhuri et al. (1999). They also 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.

Identifying the optimal query plan during query optimization may require executing several candidate query plans on the sample. A standard bottom-up candidate plan generation allows us to execute sub-plans once, store their results and reuse them multiple times as they will be common to many candidate plans. While the overhead of this execution-based selectivity estimation approach will still likely be higher than that of pre-computation based techniques (e.g., histograms), the reduced execution times of highly optimized plans enabled by better estimates, especially for complex and long-running queries, will more than compensate for this overhead. Thus, storing intermediate results that are common to several executions will speed up the total execution time on the sample. The significant improvement in the selectivity estimates in complex queries well compensates for the extra work in computing the selectivity estimates.

6. Experiments

This section presents the results of the experiments we run to validate our theoretical results and to compare our selectivity estimation method with standard techniques implemented in PostgreSQL and in Microsoft SQL Server.

Goals. 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 random samples

of different sizes. We use the selectivity of the each query in the random samples as an estimator for the selectivity in the large database with the adjustments for join operations, as described in the previous Section. We compute the error between the estimate and the actual selectivity to show that the thesis of Thm. 12 is indeed valid in practice. The use of a large number of queries and of a variety of parameters allows us to evaluate the error rate as a function of the sample size. We then compare our method with the commonly used selectivity estimation based on precomputed histograms (briefly described in Sect. 6.1). We use histograms with a different number of buckets to show that, no matter how fine-grained the histograms might be, as soon as the inter-column and intra-bucket assumptions are no longer satisfied, our approach gives better selectivity estimates.

6.1 Selectivity Estimation with Histograms

In many modern database systems, the query optimizer relies on histograms for computing data distribution statistics to help determine the most efficient query plans. In particular, PostgreSQL uses one-dimensional equi-depth (i.e., equal frequency buckets) histograms and a list of the most common values (MCV) for each column (of a database table) to compute optimizer statistics. The MCV information stores the most frequent N items (by default $N = 100$) and their frequency for each column. The histograms (by default with 100 bins) are built for the values not stored in the MCV list. The selectivity of a constraint $A = x$, where A is a column and x is a value is computed from the MCV list if x is in the MCV list or from the histogram bin that contains x if x is not in the MCV list. The selectivity of a range constraint such as $A < x$ is computed with information from both the MCV list and the histogram, i.e., the frequencies of the most common values less than x and the frequency estimate for $A < x$ from the histogram will be added to obtain the selectivity.

In PostgreSQL, the histograms and the MCV lists for the columns of a table are built using a random sample of the tuples of the table. The histograms and the MCV list for all columns of a table are based on the same sample tuples (and are therefore correlated). The sample size is computed for each column using a formula based on the table size, histogram size, and a target error probability developed by Chaudhuri et al. (1998) and the largest sample size required by the columns of a table is used to set the sample size of the table.

Finally, the join selectivity of multiple constraints are computed using the attribute independence assumption: e.g., selectivities are added in case of an OR operator and multiplied for an AND operator. Therefore, large selectivity estimation errors are possible for complex queries and correlated inputs.

6.2 Setup

Original tables. The tables in our large database were randomly generated and contain 20 million tuples each. There is a distinction between tables used for running selection queries and tables used for running join (and selection) queries. For tables on which we run selection queries only, the distributions of values in the columns fall in two different categories:

- **Uniform and Independent:** The values in the columns are chosen uniformly and independently at random from a fixed domain (the integer interval $[0, 200000]$, the same for all columns). Each column is treated independently from the others.

- **Correlated:** Two columns of the tables contain values following a multivariate normal distribution with mean $M = \mu \mathbb{I}_{2,2}$ and a non-identity covariance matrix Σ (i.e., the values in the two different columns are correlated).

The tables for join queries should be considered in pairs (A, B) (i.e., the join happens along a common column C of tables A and B). The values in the columns are chosen uniformly and independently at random from a fixed domain (the integer interval $[0, 200000]$, the same for all columns). Each column is treated independently from the others.

Sample tables. 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 drew random tuples uniformly at random from the base tables independently and added a column *sampleindex* to each tuple such that each tuple drawn from the same base table has a different value in the additional column and with tuples from different tables forming an element of the sample (of the Cartesian product of the base tables) if they have the same value in this additional column, as described in Sect. 5.2.

For each table in the original database we create many sample tables of different sizes. The sizes are either fixed arbitrarily or computed using (1) from Thm. 12. The arbitrarily sized sample tables contain between 10000 and 1.5 million tuples. To compute the VC-dimension-dependent sample size, we fixed $\varepsilon = 0.05$, $\delta = 0.05$, and $c = 0.5$. 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. If we let m be the number of columns involved in the selection predicate of the queries and b be the number of Boolean clauses in the predicate, we have that d depends directly on m and b, as does the sample size s through (1) in Thm. 12. For selection queries, we used $m = 1, 2$ and $b = 1, 2, 3, 5, 8$, with the addition of the combination $m = 5$, $b = 5$. We run experiments on join queries only for some combinations of m and b (i.e., for $m = 1$ and $b = 1, 2, 5, 8$) due to the large size of the resulting sample tables. Table 2 shows the sample sizes, as number of tuples, for the combinations of parameters we used in our experiments.

Histograms. We built histograms with a different number of buckets, ranging from 100 to 10000. Due to limitations in PostgreSQL, incrementing the number of buckets in the histograms also increments the number of values stored in the MCV list. Even if this fact should have a positive influence on the quality of the selectivity estimates obtained from the histograms, our results show that the impact is minimal, especially when the inter-column independence and the intra-bucket uniformity assumptions are not satisfied. 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.

Queries. 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 involving m columns and b Boolean clauses. The parameters in each clause, the range quantifiers, and the Boolean operators connecting the different clauses were chosen uniformly at random to ensure a wide coverage of possible queries.

		Query type			
		Select		Join	
Columns (m)	Boolean clauses (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		

Table 2: VC-Dimension bounds and samples sizes, as number of tuples, for the combinations of parameters we used in our experiments.

6.3 Results

Selection Queries. The first result of our experiments is that, for all the queries we ran, on all the sample tables, the estimate of the selectivity computed using our method 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 values in the columns, the default PostgreSQL histograms predicted a selectivity more than ε off from the real selectivity for 30 out of 100 queries. Nevertheless, in some of cases the histograms predicted a selectivity closer to the actual one than what our method predicted. This is especially true when the histogram independence assumption holds (e.g., for $m = 2$, $b = 5$ the default histograms gave a better prediction than our technique in 11 out of 100 cases). Similar situations also arise for SQLServer.

Since the selectivity estimated by the our method was always within ε from the actual, we report the actual percent error, i.e., 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 analyze the average and the standard deviation of this quantity on a set of queries and the evolution of these measures as the sample size increases. We can see from Fig. 2 and 3 that both the average and the standard deviation of the percentage error of the prediction obtained with our method decrease as the sample size grows (the rightmost plotted sample size is the one from Table 2, i.e., the one computed in Thm.12. More interesting is the comparison in those figures between the performance of the histograms and the performance of our techniques in predicting selectivities. When the assumptions of the histograms hold, as is the case for the data plotted in Fig. 2, the predictions obtained from the histograms are good.

But as soon as the data are correlated (Fig. 3), our sampling method gives better predictions than the histograms even at the smallest sample sizes and keeps improving as

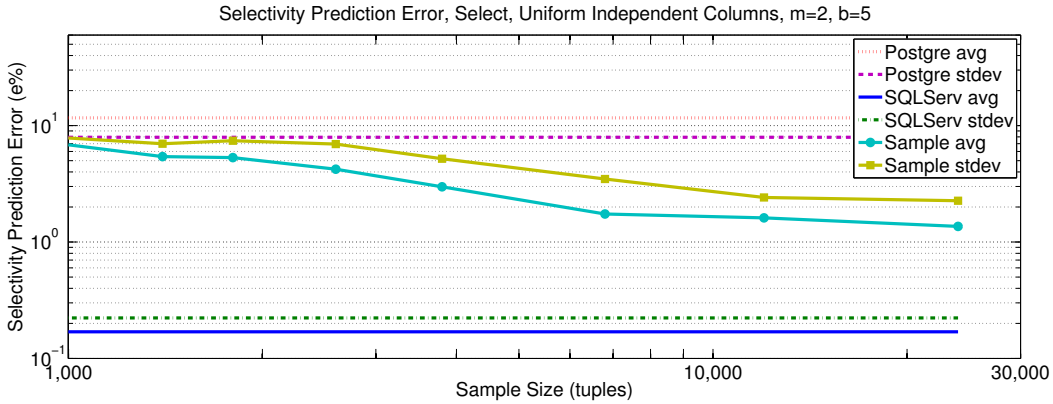

Figure 2: Selectivity prediction error for selection queries on a table with uniform independent columns – Two columns ($m = 2$), five Boolean clauses ($b = 5$).

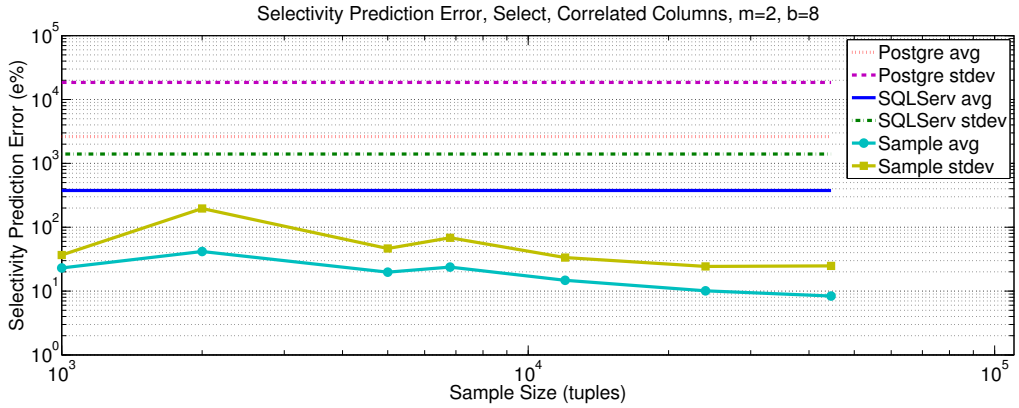

Figure 3: Selectivity prediction error for selection queries on a table with correlated columns – Two columns ($m = 2$), eight Boolean clauses ($b = 8$).

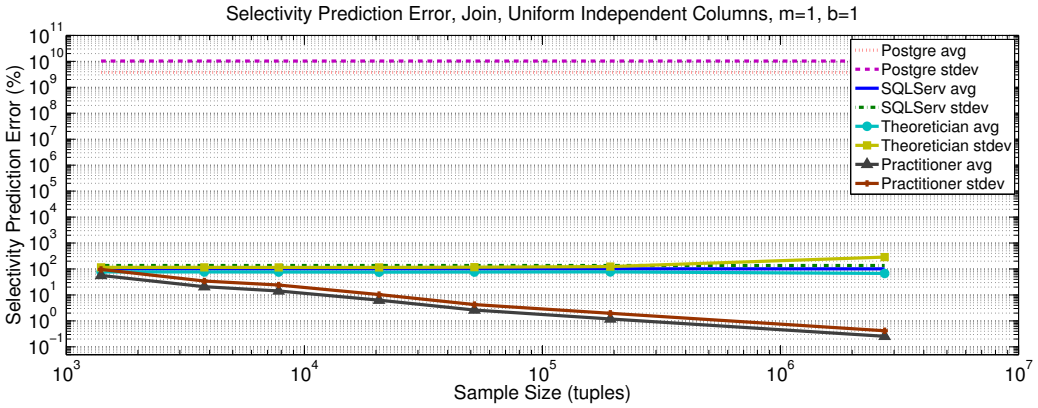

Figure 4: Selectivity prediction error for join queries queries. One column ($m = 1$), one Boolean clause ($b = 1$).

the sample grows larger. It is also interesting to observe how the standard deviation of the prediction error is much smaller for our method than for the histograms, suggesting a much higher consistency in the quality of the predictions. In Fig. 3 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, sometimes even in the negative sense (i.e., an histogram with more buckets gave worse predictions than an histogram with fewer 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: even when the multi-column statistics were supposed to help, as in the case of correlated data, the obtained prediction were not much different from the ones obtained from the single-column histograms.

Join Queries. The strength of our method compared to histograms is even more evident when we run join queries, even when the histograms independent assumptions are satisfied. In our experiments, the predictions obtained using our technique were always within ε from the real values, even at the smallest sample sizes, but the same was not true for histograms. For example, in the case of $m = 1$ and $b = 5$, 135 out of 300 predictions from the histograms were more than ε off from the real selectivities. Figure 4 shows the comparison between the average and the standard deviation of the percentage error, defined in the previous paragraph, for the histograms and our method. The numbers include predictions for the selection operations at the leaves of the query tree.

Again, we did not plot multiple curves for histograms with a different number of buckets because the quality of the predictions did not improve as the histograms became more fine-grained. To understand the big discrepancy between the accurate predictions of our method and the wrong estimates computed by the histograms in PostgreSQL we note that for some join queries, the histograms predicted an output size on the order of the hundreds of thousands of tuples but the actual output size was zero or a very small number of tuples. Observing the curves of the average and the standard deviation of the percentage error for

the prediction obtained with our method, we can see that at the smaller sample sizes the quality of the predictions only improves minimally with the sample size. This is due to the fact that at small sizes our prediction for the join operation is very often zero or very close to zero, because the output of the query does not contain enough pairs of tuples from the sample of the Cartesian product of the input table (i.e., pairs of tuples with the same value in the *sampleindex* column). In these cases, the prediction can not be accurate at all (i.e., the error is 100% if the original output contained some tuples, or 0% if the query returned an empty set in the large databases). As soon as the sample size grows more, we can see first a jump to higher values of the percentage error, which then behaves as expected, i.e., decreasing as the sample size increases.

In Fig. 4 we also show a comparison between the percentage error of predictions obtained using our method 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 and the "practitioner" way which uses the size of the output of the join operation in the sample, therefore ignoring the *sampleindex* column. Recall that we had to add the *sampleindex* column because Thm. 12 requires a uniform sample of the Cartesian product of the input tables. As it is evident from Fig. 4, the "practitioner" way of predicting selectivity gives very good results at small sample sizes (although it does not offer theoretical guarantees). These results are similar in spirit, although not equivalent, to the theoretical conclusions presented by Haas et al. (1996) in the setting of selectivity estimation using online sampling.

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 multi-dimensional histograms (Bruno et al., 2001; Poosala and Ioannidis, 1997; Srivastava et al., 2006; Wang and Sevcik, 2003) and join synopses (Acharya et al., 1999). The main advantage of our method is that it gives uniformly accurate estimates for the selectivity of any query within a predefined VC-dimension range. Methods that collect and store pre-computed statistics give accurate estimates only for the relations captured by the collected statistics, while estimate of any other relation 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.

Acknowledgements

This work was supported in part by the National Science Foundation, under grant IIS-0905553.

References

Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. *SIGMOD Rec.*, 28:275–286, June 1999.

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.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1999.

Brian Babcock, Surajit Chaudhuri, and Gautam Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 539–550, New York, NY, USA, 2003. ACM.

Michael Benedikt and Leonid Libkin. Aggregate operators in constraint query languages. *Journal of Computer and System Sciences*, 64(3):628–654, 2002.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 609–618, New York, NY, USA, 2008. ACM.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36:929–965, October 1989.

Paul G. Brown and Peter J. Haas. Techniques for warehousing of sample data. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, page 6, Washington, DC, USA, 2006. IEEE Computer Society.

Nicolas Bruno and Surajit Chaudhuri. Conditional selectivity for statistics on query expressions. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 311–322, New York, NY, USA, 2004. ACM.

Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. STHoles: a multidimensional workload-aware histogram. *SIGMOD Rec.*, 30:211–222, May 2001.

Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. Random sampling for histogram construction: how much is enough? *SIGMOD Rec.*, 27:436–447, June 1998.

Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. On random sampling over joins. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 263–274, New York, NY, USA, 1999. ACM.

Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32:50, June 2007.

Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000.

Meng Chang Chen, Lawrence McNamee, and Norman S. Matloff. Selectivity estimation using homogeneity measurement. In *Proceedings of the Sixth International Conference on Data Engineering*, pages 304–310, Washington, DC, USA, 1990. IEEE Computer Society.

Gautam Das. Sampling methods in approximate query answering systems. In John Wang, editor, *Encyclopedia of Data Warehousing and Mining*, pages 1702–1707. IGI Global, Hershey, PA, USA, 2nd edition, 2009.

Alin Dobra. Histograms revisited: when are histograms the best approximation method for aggregates over joins? In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 228–237, New York, NY, USA, 2005. ACM.

Cristian Estan and Jeffrey F. Naughton. End-biased samples for join cardinality estimation. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 20–, Washington, DC, USA, 2006. IEEE Computer Society.

Sumit Ganguly, Phillip B. Gibbons, Yossi Matias, and Avi Silberschatz. Bifocal sampling for skew-resistant join size estimation. *SIGMOD Rec.*, 25:271–281, June 1996.

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.

Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems - The Complete Book*. Prentice Hall, Upper Saddle River, NJ, USA, 2002.

Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. A dip in the reservoir: maintaining sample synopses of evolving datasets. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 595–606, Almaden, CA, USA, 2006. VLDB Endowment.

Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. Maintaining bernoulli samples over evolving multisets. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, pages 93–102, New York, NY, USA, 2007. ACM.

Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. *SIGMOD Rec.*, 30:461–472, May 2001.

Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. *SIGMOD Rec.*, 27:331–342, June 1998.

David Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. *ACM Trans. Database Syst.*, 36:3:1–3:24, March 2011.

Jarek Gryz and Dongming Liang. Query selectivity estimation via data mining. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'04 Conference held in Zakopane, Poland, May 17-20, 2004*, Advances in Soft Computing, pages 29–38, Berlin Heidelberg, Germany, 2004. Springer-Verlag.

Peter J. Haas. Hoeffding inequalities for join-selectivity estimation and online aggregation. Technical Report RJ 10040, IBM Almaden Research, 1996.

Peter J. Haas and Christian König. A bi-level Bernoulli scheme for database sampling. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 275–286, New York, NY, USA, 2004. ACM.

Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, SIGMOD '92, pages 341–350, New York, NY, USA, 1992. ACM.

Peter J. Haas and Arun N. Swami. Sampling-based selectivity estimation for joins using augmented frequent value statistics. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 522–531, Washington, DC, USA, 1995. IEEE Computer Society.

Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. Fixed-precision estimation of join selectivity. In *Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '93, pages 190–201, New York, NY, USA, 1993. ACM.

Peter J. Haas, Jeffrey F. Naughton, and Arun N. Swami. On the relative cost of sampling for join selectivity estimation. In *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '94, pages 14–24, New York, NY, USA, 1994. ACM.

Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52:550–569, June 1996.

Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

Banchong Harangsri, John Shepherd, and Anne H. H. Ngu. Query size estimation using machine learning. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 97–106, Hackensack, NJ, USA, 1997. World Scientific Press.

D Haussler and E Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the second annual symposium on Computational geometry*, SCG '86, pages 61–71, New York, NY, USA, 1986. ACM.

Wen-Chi Hou, Gultekin Ozsoyoglu, and Baldeo K. Taneja. Statistical estimators for relational algebra expressions. In *Proceedings of the seventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '88, pages 276–287, New York, NY, USA, 1988. ACM.

Wen-Chi Hou, Gultekin Ozsoyoglu, and Erdogan Dogdu. Error-constrained count query evaluation in relational databases. *SIGMOD Rec.*, 20:278–287, April 1991.

Yannis E. Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, pages 233–244, New York, NY, USA, 1995. ACM.

H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 275–286, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

Christopher Jermaine, Abhijit Pol, and Subramanian Arumugam. Online maintenance of very large random samples. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 299–310, New York, NY, USA, 2004. ACM.

Ruoming Jin, Leo Glimcher, Chris Jermaine, and Gagan Agrawal. New sampling-based estimators for olap queries. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 18–, Washington, DC, USA, 2006. IEEE Computer Society.

Shantanu Joshi and Christopher Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 199–208, Washington, DC, USA, 2008. IEEE Computer Society.

Raghav Kaushik, Jeffrey F. Naughton, Raghu Ramakrishnan, and Venkatesan T. Chakravarthy. Synopses for query optimization: A space-complexity perspective. *ACM Trans. Database Syst.*, 30:1102–1127, December 2005.

Per-Ake Larson, Wolfgang Lehner, Jingren Zhou, and Peter Zabback. Cardinality estimation using sample views with quality assurance. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 175–186, New York, NY, USA, 2007. ACM.

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.

Richard J. Lipton and Jeffrey F. Naughton. Query size estimation by adaptive sampling. *J. Comput. Syst. Sci.*, 51:18–25, August 1995.

Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. *SIGMOD Rec.*, 19:1–11, May 1990.

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, Berlin / Heidelberg, 2009.

V. Markl, Peter. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tran. Consistent selectivity estimation via maximum entropy. *The VLDB Journal*, 16:55–76, January 2007.

Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 448–459, New York, NY, USA, 1998. ACM.

Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, Secaucus, NJ, USA, 2002.

Rupert G. Miller. *Simultaneous Statistical Inference*. Springer series in statistics. Springer-Verlag, New York, NY, USA, 1981.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.

A. H. H. Ngu, B. Harangsri, and J. Shepherd. Query size estimation for joins using systematic sampling. *Distrib. Parallel Databases*, 15:237–275, May 2004.

Frank Olken. *Random Sampling from Databases*. PhD thesis, University of California, Berkeley, 1993.

Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 486–495, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, SIGMOD '96, pages 294–305, New York, NY, USA, 1996. ACM.

Christopher Ré and Dan Suciu. Understanding cardinality estimation using entropy maximization. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '10, pages 53–64, New York, NY, USA, 2010. ACM.

Matteo Riondato and Eli Upfal. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I*, volume 7523 of *Lecture Notes in Computer Science*, pages 25–41, Berlin / Heidelberg, 2012. Springer.

U. Srivastava, Peter J. Haas, V. Markl, M. Kutsch, and T. M. Tran. ISOMER: Consistent histogram construction using query feedback. In *Proceedings of the 22nd International*

Conference on Data Engineering, ICDE '06, pages 39–, Washington, DC, USA, 2006. IEEE Computer Society.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for engineering and information science. Springer-Verlag, New York, NY, USA, 1999.

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.

Hai Wang and Kenneth C. Sevcik. A multi-dimensional histogram for selectivity estimation and fast approximate query answering. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '03, pages 328–342, Boston, MA, USA, 2003. IBM Press.

Min Wang, Jeffrey Scott Vitter, and Balakrishna R. Iyer. Selectivity estimation in the presence of alphanumeric correlations. In *Proceedings of the Thirteenth International Conference on Data Engineering*, ICDE '97, pages 169–180, Washington, DC, USA, 1997. IEEE Computer Society.

Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. Applying the golden rule of sampling for query estimation. *SIGMOD Rec.*, 30:449–460, May 2001.