Received March 25, 2020, accepted April 7, 2020, date of publication April 15, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988120

Sampling for Big Data Profiling: A Survey

ZHICHENG LIU[1] **AND AOQIAN ZHANG**[2]
[1]School of Software, Tsinghua University, Beijing 100084, China
[2]Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Corresponding author: Zhicheng Liu (lzc18@mails.tsinghua.edu.cn)

This work was supported in part by the National Key Research and Development Plan under Grant 2019YFB1705301, and in part by the National Natural Science Foundation of China under Grant 61572272 and Grant 71690231.

ABSTRACT Due to the development of internet technology and computer science, data is exploding at an exponential rate. Big data brings us new opportunities and challenges. On the one hand, we can analyze and mine big data to discover hidden information and get more potential value. On the other hand, the 5V characteristic of big data, especially Volume which means large amount of data, brings challenges to storage and processing. For some traditional data mining algorithms, machine learning algorithms and data profiling tasks, it is very difficult to handle such a large amount of data. The large amount of data is highly demanding hardware resources and time consuming. Sampling methods can effectively reduce the amount of data and help speed up data processing. Sampling technology has been widely used in big data context. Data profiling is the activity that finds metadata of data set and has many use cases, e.g., performing data profiling tasks on relational data, graph data, and time series data for anomaly detection and data repair. However, data profiling is computationally expensive, especially for large data sets. Hence this article focuses on researching sampling for data profiling tasks in big data context and investigates the application of sampling in different categories of data profiling. From the experimental results of these studies, the results got from the sampled data are close to or even exceed the results of the full amount of data. Therefore, sampling technology plays an important role in the era of big data, and we also have reason to believe that sampling technology will become an indispensable step in big data processing in the future.

INDEX TERMS Big data, large amount, sampling, data profiling.

I. INTRODUCTION

We are in the era of big data. With the development of computer science and internet technology, data is exploding at an exponential rate. According to statistics, Google processes more than hundreds of PB data per day, Facebook generates more than 10 PB of log data per month, Baidu processes nearly 100 PB of data per day, and Taobao generates dozens of terabytes online transaction data every day [1]. In May 2011, the McKinsey Global Institution (MGI) released the report[1] which said that big data has great potential in the European Public Sector, US Health Care, Manufacturing, US Retail Industry and Location-based Services. MGI estimates in the report that the mining and analysis of big data will generate 300 billion in potential value per year in the US medical sector and more than 149 billion in the European public sector [2]. It can be seen that there is great value behind big data. Therefore, mining the hidden value under big data makes a lot of sense.

Big data is something so huge and complex that it is difficult or impossible for traditional systems and tools to process and work on it [3]. In the latest development, IBM uses "5Vs" model to depict big data. In the "5Vs" model, Volume means the amount of data and it is the most direct difficulty faced by traditional systems; Velocity means that data is generated quickly; Variety means that data sources and data types are diverse including structural, semi-structured, and unstructured data; Value is the most important feature of big data, although the value density of data is low; Veracity refers to that data quality of big data where there is dirty data. Because big data is so large that data analysis and data mining based on big data require high computing power and storage capacity. In addition, some classical mining algorithms that require several passes over the whole dataset may take hours or even days to get result [4].

A. DATA SAMPLING

At present, there are two major strategies for data mining and data analysis: sampling and using distributed systems [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Le Hoang Son.

[1]The Next Frontier of Big Data: Innovation, Competition, and Productivity

The existing big data processing framework includes batch processing framework like Apache Hadoop, streaming data processing framework like Apache Storm, hybrid processing framework like Apache Spark and Apache Flink. Sampling is a scientific method of selecting representative sample data from target data. Designing a big data sampling mechanism is to reduce the amount of data to a manageable size for processing [6]. Even if computer clusters are available, we can use sampling such as block-level sampling to speed up big data analysis [7].

Different from distributed systems, sampling is a kind of data reduction method like filtering. Distributed systems increase computing power by adding hardware resources. However, a huge computing cost is not always affordable in practice. It is highly demanded to perform the computing under limited resources. In this sense, sampling is very useful. Since the full amount of data is not used, the approximate result is obtained from the sample data. Such approximate result is quite useful in the context of big data. The computational challenge of big data means that sampling is essential and the sampling methods chosen by researchers is also important [8]. Besides, the biases caused by sampling are also something need to be considered.

Sampling or re-sampling is to use less data to get the overall characteristics of the whole dataset. Albattah [9] studies the role of sampling in big data analysis. He believes that even if we can handle the full amount of data, we don't have to do this. They focus on how sampling will play its role in specific fields of Artificial Intelligence and verify it by doing experiments. The experimental results show that sampling not only reduces the data processing time, but also get better results in some cases. Even though some examples of sampling are not as effective as the original dataset, they are obviously negligible compared to the greatly reduced processing time. As stated in [9], we believe that sampling can improve big data analysis and will become a preprocessing step in big data processing in the future.

B. DATA PROFILING

Data mining is an emerging research area, whose goal is to extract significant patterns or interesting rules from large data sets [10]. Data profiling gathers metadata of data that can be used to find data to be mined and import data into various tools for analysis, which is an important preparatory task [11]. There is currently no formal, universal or widely accepted definition of distinction between data profiling and data mining. Abedjan *et al.* [12] think data profiling is used to generate metadata for data sets that are used to help understand data sets and manage data sets. However, data mining is used to mine the hidden knowledge behind the data, which is not so obvious. Of course, data profiling and data mining also have some overlapping tasks, such as association rule mining and clustering. In summary, the goal of data profiling is to generate summary information about the data to help understand the data, and the goal of data mining is to mine the new insights of the data.

There are many use cases of data profiling, such as data profiling for missing data imputation [13] or erroneous data repairing in relational database [14]. However, data profiling itself has to face computational challenges, especially when it comes to large data sets. Hence how to alleviate the computational challenges of data profiling is very significant in era of big data. As mentioned above, sampling for big data profiling is very valuable and meaningful.

C. SAMPLING FOR DATA PROFILING

In this paper, we focus on the sampling techniques used for big data profiling. Certainly, we will first introduce data profiling and sampling technology separately. Among them, data profiling has been associated with outstanding survey papers such as [12]. Finally, our core content is to introduce the application of sampling in data profiling tasks when facing large data sets.

In [12], the research on data profiling around the relational database is fully investigated and introduced. The classification of data profiling (see Figure 1) is given in [12]. We will investigate the sampling techniques for important data profiling tasks in single column, multiple columns and dependency according to the classification of data profiling in [12]. Some traditional sampling methods are introduced in [15], and methods of determining the sample size are mainly introduced, but less attention is paid to sampling in big data context. Therefore, when discussing the sampling technology below, we will supplement some applications and information of sampling in the big data scenario, e.g., block-based sampling.

Specifically, in order to ensure the comprehensiveness of the survey, we follow the systematic search method provided in [12], a comprehensive summary of data profiling techniques. As also illustrated in Figure 1 of our manuscript, Abedjan *et al.* [12] categorize the data profiling approaches into three aspects, from the elementary columns to the complex ones, i.e., (1) data profiling for single columns, (2) data profiling for multiple columns, and (3) data profiling for dependencies. While the sampling techniques for data profiling are not emphasized in [12], in our paper, we extensively select the studies on sampling for data profiling in the aforesaid categories, respectively. Figure 2 presents the systematic search method for selecting studies, following the categorization in [13]. Following this method, we summarize the typical methods selected in each category in Table 3.

The remaining of this paper is organized as follows. In Section II, we introduce the relevant knowledge of data profiling. Besides, we introduce sampling techniques and some important factors in sampling techniques. Next we introduce the application of sampling for single-column data profiling tasks in Section III, multi-column data profiling tasks in Section IV and dependencies in Section V based on the classification of data profiling tasks in [12]. Finally, in Section VI, we summarize the content of the article and propose some future works. The organizational structure of this article is shown in Figure 2.

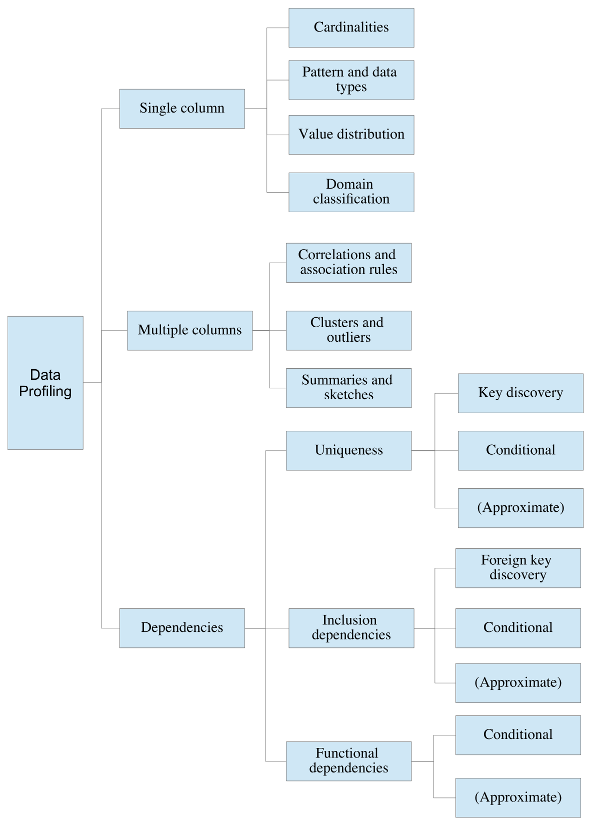


FIGURE 1. A classification of typical data profiling tasks [12].

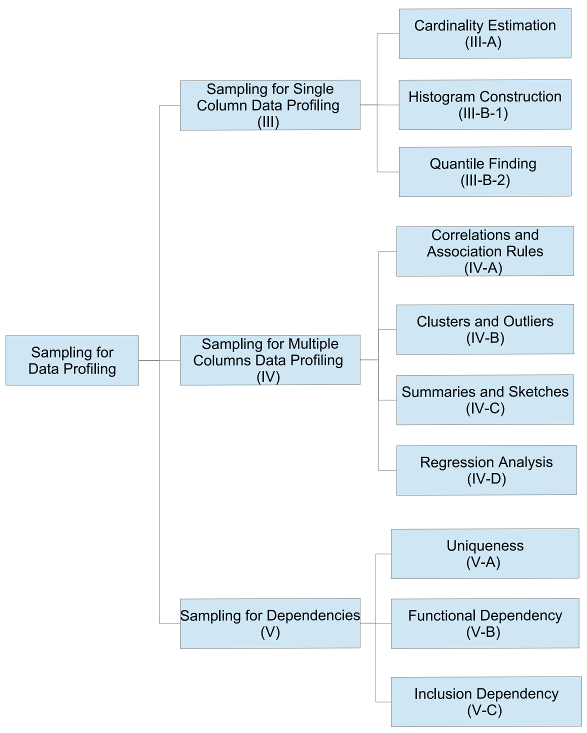

Sampling for Data Profiling
Sampling for Single Column Data Profiling (III)
Cardinality Estimation (III-A)
Histogram Construction (III-B-1)
Quantile Finding (III-B-2)
Sampling for Multiple Columns Data Profiling (IV)
Correlations and Association Rules (IV-A)
Clusters and Outliers (IV-B)
Summaries and Sketches (IV-C)
Regression Analysis (IV-D)
Sampling for Dependencies (V)
Uniqueness (V-A)
Functional Dependency (V-B)
Inclusion Dependency (V-C)

FIGURE 2. A systematic search method for selecting studies, following the categorization in Figure 1 by [12].

II. PRELIMINARIES

In this section, we introduce data profiling and sampling. For data profiling, we introduce its definition, classification and application. For sampling technology, we introduce some common sampling techniques, how to determine the sample size and how to solve the sampling bias.

A. DATA PROFILING

Before using or processing data, it is very important to have a general understanding of the data. Data profiling is the activity that finds metadata of data set [12], [16], [17], therefore it can provide basic information about data to help people understand the data. Data profiling is an important area of research for many IT experts and scholars. Data profiling has many classic use cases, such as data integration, data quality, data cleansing, big data analysis, database management, query optimization [12], [16]. Abedjan *et al.* [12] mainly investigates data profiling for relational data. However, in addition to relational databases, many non-relational databases need data profiling [16], such as time series data [18]–[20], graph data [21]–[23], or heterogeneous data in dataspaces [24]–[26].

Data profiling tasks are classified in [12] and [16]. Abedjan *et al.* [12] classify the data profiling tasks of single data source, and divides the tasks of data profiling into single column data profiling, multiple columns data profiling and dependency (see Figure 1). In fact, dependencies belong to multiple columns data profiling tasks. Abedjan *et al.* [12] put dependencies separately into a large category and discuss it in detail. Naumann [16] classifies data profiling from single data source to multiple data sources.

There are three challenges for data profiling: managing the input, performing the computation and managing the output [12], [16], [27]. In this article we focus on the second challenge, performing the computation, i.e., the computational complexity of data profiling. The computational complexity of data profiling depends on the number of rows and columns of data. When the data set is very large, the calculation of data profiling can be very expensive. This is why we care about sampling for big data profiling, in order to reduce the computational pressure and speed up the process of data profiling.

B. SAMPLING TECHNIQUES

In this section, we introduce common sampling techniques, application of sampling technology in big data context, methods of determining sample size, sampling error and sampling bias.

Sampling refers to estimating the characteristics of the entire population through the representative subsets within the population [15]. From a big perspective, sampling involves probability and non-probability sampling. Probability sampling means that every unit in a finite population has a certain probability to be selected, and it does not necessarily require equality. Non-probability sampling is generally based on subjective ideas and inferences, e.g., common web

TABLE 1. **Common sampling methods.**

Sampling method	Description
Simple random sampling	Extracting a certain number of samples and each tuple is selected with equal probability.
Stratified random sampling	Tuples are divided into homogeneous groups and sample from each group.
Systematic sampling	Sampling at regular intervals until the sample size is satisfied.
Cluster sampling	Tuples are divided into non-overlapping groups and randomly select some groups as samples.
Oversampling and Undersampling	Oversampling randomly repeat the minority class samples, while Undersampling randomly discard the majority class samples to balance the data.
Reservoir sampling	Adding tuples into the reservoir of a fixed size with unknown size of the entire data set.

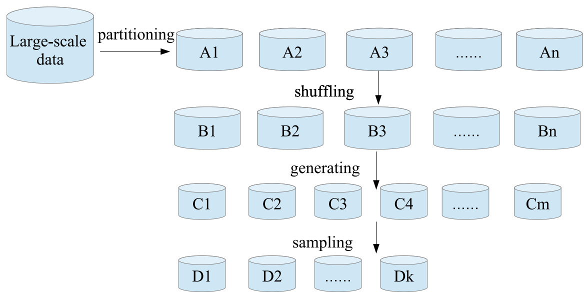


FIGURE 3. **I-sampling workflow [41].**

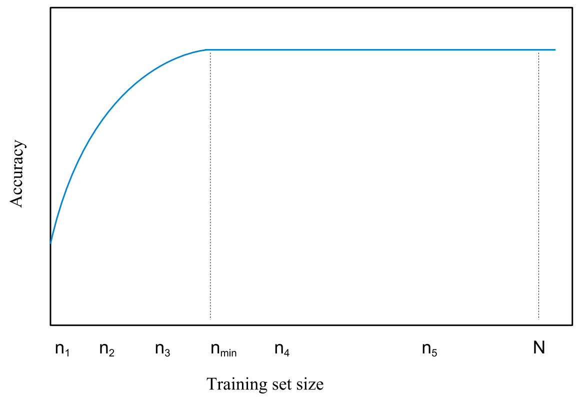


FIGURE 4. **Learning curves [44].**

questionnaires [28], [29]. The sampling methods mentioned below are all probability sampling methods. Sampling is often used in data profiling [12], data analysis [30], data mining [6], data visualization [31], machine learning [32] etc. The advantage of sampling is that algorithms or models can be conducted using subset instead of the whole data set. There are some commonly used sampling techniques including simple random sampling [33], stratified sampling [34], systematic sampling [35], cluster sampling [36], oversampling and undersampling [37], [38], reservoir sampling [39], etc. Table 1 gives an overview of these common sampling methods.

In the era of big data, the application of sampling is particularly important due to the large amount of big data. And sampling can be performed with the help of big data computing framework. For example, He *et al.* [40] use MapReduce to sample from the data which contains uncertainty. He *et al.* [41] propose a block-based sampling (I-sampling) method for large-scale dataset when the whole dataset is already assigned on a distributed system. The processing flow of I-sampling is shown in Figure 3.

It is very important to select effective samples [42]. If the sample size is too small, it may get an incorrect conclusion. If the sample size is too large, the calculation time is too long. Singh and Masuku [15] have summarized some traditional methods for determining sample size in detail. When sampling is used in machine learning, the most appropriate number of samples is to make the accuracy rate reach the maximum value and increasing the number of samples can no longer improve the accuracy of the learning algorithm. The corresponding figure is Figure 4, where n_{min} is the minimum sample size. In this case, John and Langley [43] propose a sequential sampling method called Arithmetic Sampling to find the minimum sample size.

Sampling error is when a randomly chosen sample does not reflect the underlying population purely by chance and sampling bias is when the sample is not randomly chosen at all [45]. Sampling bias is one of the causes of sampling error. These two are often confused by some scholars. Sampling bias is caused by the failure of the sampling design, which cannot truly extract the sample randomly from the population [46]. And big data is susceptible to selection bias, thereby many scholars study how to solve selection bias in sampling process [47]–[50].

III. SAMPLING FOR SINGLE COLUMN DATA PROFILING

Single column data profiling tasks are divided into cardinalities, value distributions, patterns, data types, and domains [71]. Table 2 [12] lists typical metadata that may result from single-column data profiling. For some single-column data profiling tasks, such as decimals which calculates maximum number of decimals in numeric values, simple sampling methods cannot guarantee reliable results. And for identifying a domain of one column, it is often more difficult and not fully automated [72]. Among them, cardinality, histograms and quantiles are often used for query optimizers, therefore sampling techniques are more commonly used in these tasks. Specifically, in Section III-A, we introduce sampling for cardinality estimation. Section III-B presents sampling for value distribution. More advanced statistics include the probabilistic correlations on text attributes [73].

TABLE 2. Overview of single-column profiling tasks [12].

Category	Task	Description
Cardinalities	num-rows	Number of rows
	value length	Measurements of value lengths (minimum, maximum, median, and average)
	null values	Number or percentage of null values
	distinct	Number of distinct values; sometimes called "cardinality"
	uniqueness	Number of distinct values divided by the number of rows
Value distributions	histogram	Frequency histograms (equi-width, equi-depth, etc.)
	constancy	Frequency of most frequent value divided by number of rows
	quartiles	Three points that divide the (numeric) values into four equal groups
	first digit	Distribution of first digit in numeric values
Patterns, data types, and domains	basic type	Generic data type, such as numeric, alphabetic, alphanumeric, date, time
	data type	Concrete DBMS-specific data type, such as varchar, timestamp.
	size	Maximum number of digits in numeric values
	decimals	Maximum number of decimals in numeric values
	patterns	Histogram of value patterns (Aa9...)
	data class	Semantic, generic data type, such as code, indicator, text, date/time, quantity, identifier
	domain	Classification of semantic domain, such as credit card, first name, city, phenotype

TABLE 3. Summary of sampling for big data profiling tasks.

Data Profiling		Sampling-based method
Single column	Cardinality Estimation	$\hat{D}_{hybrid}$ [51], GEE [52], AE [52], Distinct sampling [53]
	Histograms	Random sampling [54], Backing sample [55]
	Quantiles	Non-uniform random sampling [56], Improved random sampling [57]
Multiple columns	Correlations and association rules	Sequential random sampling without replacement [10], Two-phased sampling [4], ISbFIM [58]
	Clusters and outliers	Biased sampling [59]
	Summaries and sketches	Error-bounded stratified sampling [60], [61]
	Regression analysis	IBOSS [62], Random sampling without replacement [63]
Dependency	Uniqueness	GORDIAN [64], HCA-Gordian [65]
	Functional dependencies	AID-FD [66], HYFD [67], CORDS [68], BRRSC [69]
	Inclusion dependencies	FAIDA [70]

A. SAMPLING FOR CARDINALITY ESTIMATION

Cardinalities or counts of values in a column are the most basic form of metadata [12]. Cardinalities usually include number of rows, number of null values and number of distinct values, which is the most important type of metadata [74]. For some tasks, such as number of rows and number of null values, a single pass over a column can get the exact result. However, finding the number of distinct values may require to sort or hash the value of column [72]. Similarly, when facing large data sets, statistics of the number of distinct values of an attribute have to face the pressure of memory and calculation. Therefore, the estimation of the number of distinct values based on sampling have been studied [51]–[53].

Haas *et al.* [51] propose several sampling-based estimators to estimate the number of different values of an attribute in a relational database. They use a large number of attribute value distributions from various actual databases to compare these new estimators with those in databases and statistical literature. Their experimental results prove that no estimator is optimal for all attribute value distributions. And from their experimental results, it can be seen that the larger the sampling fraction, the smaller the estimated mean absolute deviation will be. They therefore propose a sampling-based hybrid estimator $\hat{D}_{hybrid}$ and get the highest precision on average at a given sampling fraction.

Similar to Haas *et al.*, Charikar *et al.* [52] also obtain a negative result in the experiment that no estimator based on sampling can guarantee small errors on the input data of different distributions, unless a larger sampling fraction is performed on the input data. They therefore propose a new estimator Guaranteed-Error Estimator (GEE), which is provably optimal. Although its error on the input of different distributions is small, it does not make use of the knowledge of different distributions. For example, in the case of low-skew data with a large number of distinct values, GEE performs not very well in practice. They further propose a new heuristic version of GEE called Adaptive Estimator (AE), which avoids the problems encountered by GEE.

Different from the previous research using random sampling, Gibbons [53] proposes distinct sampling to accurately estimate the number of distinct values. Distinct sampling can collect distinct samples in a single scan of the data, and the samples can be kept up to date in the state of data deletions and insertions. On a truly confidential data set Call-center, distinct sampling uses only 1% of the data, and can achieve a relative error of 1% -10%, while increasing the speed of report generation by 2-4 orders of magnitude. They compare distinct sampling with GEE, AE in the experiment and prove that in real-world data sets, distinct sampling performs much better than GEE and AE.

It is worth noting that Harmouch and Naumann [74] conduct an experimental survey on cardinality estimation. In the experiment, they use the GEE [52] as an example of evaluation. They perform experiments on synthetic and real-world data sets. It can be seen from the experimental results that the larger the sampling fraction, the smaller the average estimation relative error. And when GEE wants to reach 1% relative error, it needs to collect more than 90% of the data. In conclusion, when faced with large data sets, cardinality estimation requires high memory, and sampling can reduce memory consumption, but cannot guarantee reasonable accuracy all input distributions.

B. SAMPLING FOR VALUE DISTRIBUTION

Value distribution is a very important part of single-column data profiling. Histogram and quantile are two typical forms used to represent value distribution. The histogram is used to describe the distribution of data, while quantile refers to dividing the data into several equal parts.

1) SAMPLING FOR HISTOGRAM CONSTRUCTION

Many commercial database systems maintain histograms to summarize the contents of large relations and permit efficient estimation of query result sizes for use in query optimizers [55]. Histogram can be used to describe the frequency distribution of attributes of interest, which groups attributes values into buckets and approximates true attribute values and their frequencies based on summary statistics maintained in each bucket [75]. However, the database is updated frequently, hence the histogram also needs to be updated accordingly. Recalculating histograms is expensive and unwise for large relations.

Gibbons *et al.* [55] propose sampling-based approaches for incremental maintenance of approximate histograms. They use a "backing sample" to update histograms. Backing sample is a random sample of the relation which is kept up to date in the presence of databases updates, which is generated by uniform random sampling. Therefore, random sampling can help to speed up histogram re-computation. For example, SQL Server recomputes histograms based on a random sample from relations [54].

Chaudhuri *et al.* [54] focus on how much sample is enough to construct a histogram. They propose a new error metric called the max error metric for approximate equip-depth histogram. The max error metric is formula (1) shown below, where b_j is number of values in bucket j, k is the number of buckets and n is the number of records. A k-histogram is said to be a δ-deviant histogram when $\Delta max \leq \delta$. And size of sample r is calculated as the following formula (2), where $\delta \leq \frac{n}{k}$ and γ is predefined probability.

$$\Delta max = \max_{1 \leq j \leq k} |b_j - \frac{n}{k}| \tag{1}$$

$$r \geq \frac{4n^2 \ln(\frac{2n}{\gamma})}{k\delta^2} \tag{2}$$

2) SAMPLING FOR QUANTILE FINDING

Quantiles can be used to represent the distribution of single column value. Quantiles are used by query optimizers to provide selectivity estimates for simple predicates on table values [76]. Calculating exact quantiles on large data sets is time consuming and requires a lot of memory. For example, quantile finding algorithm in [77] requires to store at least N/2 data elements to find the median, which is memory unacceptable for large-scale data.

Therefore, Manku *et al.* [56] present a novel non-uniform random sampling to find approximate quantile. They apply non-uniform random sampling to reduce memory requirements. Non-uniform means that the probability of selecting each element in the input is different. They set the earlier elements in the input sequence with larger probability than those arrive later. And the process of quantile finding is shown in Figure 5. When the data arrives, they randomly select an element in each data block and put it into buffers. Then based on sample, deterministic algorithms are performed to find quantiles.

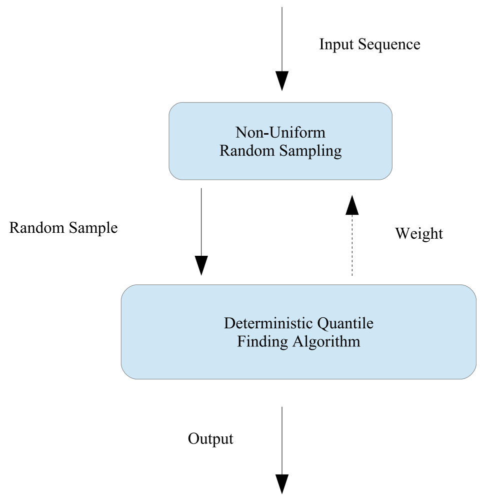

Input Sequence
Non-Uniform Random Sampling
Random Sample
Weight
Deterministic Quantile Finding Algorithm
Output

FIGURE 5. Sampling for quantile finding [56].

However, simply using random sampling method and calculating the quantiles on the sample may not be accurate enough on sensor networks. Hence Huang *et al.* [57] propose a new sampling-based quantile computation algorithm for sensor networks to reduce the communication cost. To improve accuracy, they augment the random sample with additional information about the data. They analyze how to add additional information to the random sample under the flat model and the tree model. For example, in the flat model, each node first samples each data value independently with a certain probability p and computes its local rank. Then the samples and their local ranks are sent to base station. The base station estimates rank for any value it receives and then quantile queries can be solved. In the end, they prove through experiments that the quantile computation in Sensor Networks based on this new sampling method reduces one to two orders of magnitude in terms of the total communication cost compared with the previous method.

IV. SAMPLING FOR MULTIPLE COLUMNS DATA PROFILING

As shown in Figure 1, the content of the multiple columns data profiling tasks includes association rule mining [78], clusters and outliers [79], summaries and sketches [12]. Besides, statistical methods such as regression analysis [80] can be used to perform multiple columns analysis, analyzing the relationship between these columns. Specifically, in Section IV-A, we investigate sampling for discovering association rules. Section IV-B presents the content of sampling for clusters and outliers. And sampling for summaries and sketches is introduced in Section IV-C. Then, in Section IV-D, we introduce sampling for helping perform regression analysis.

A. SAMPLING FOR DISCOVERING ASSOCIATION RULES

The discovery of association rules is a typical problem in data profiling for multiple columns. The algorithm currently used to find association rules needs to scan the database several times. For large data sets, the time overhead of scanning several times is hard to accept. Large amount of data leads to input data, intermediate results and output patterns can be too large to fit into memory and prevents many algorithms from executing [58]. Some scholars have proposed using parallel or distributed methods to solve the problem of data volume [81], [82]. But it is difficult to design parallel or distributed algorithms.

Therefore, Zaki *et al.* [10] use sampling to get samples of transaction and find the association rules based on the obtained samples. They take sequential random sampling without replacement as their sampling method and use Chernoff bounds to obtain sample size. Finally, they experimentally prove that sampling can speed up the discovery of association rules by more than an order of magnitude and provide high accuracy for association rules.

Chen *et al.* [4] propose a two-phased sampling-based algorithm to discover association rules in large databases. At the first stage, a large initial sample of transactions is randomly selected from databases, which is applied to calculate support of each individual item. And these estimated supports are used to trim the initial sample to a smaller final sample S0. At the second stage, association-rule algorithm is performed against the final sample S0 to get association rules according to provided minimum support and confidence. In the experiment, the authors prove 90-95% accuracy obtained using the final sample S0 and the size of sample is only 15-33% of the whole databases. This again proves that sampling can be used to speed up data analysis and big data profiling.

Wu *et al.* [58] propose an Iterative Sampling based Frequent Itemset Mining method called ISbFIM. The same as [10], Wu *et al.* [58] use random sampling as the sampling method. But the difference is that they use iterative sampling to get multiple subsets and find frequent items from these subsets. They can guarantee that the most frequent patterns for the entire data set have been enumerated and implement a Map-Reduce version of ISbFIM to demonstrate its scalability on big data. Because the volume of input data is reduced, the problem that input data, intermediate results, or the final frequent items cannot be loaded into memory is solved. And the traditional exhaustive search-based algorithms like Apriori can be fitted for big data context.

B. SAMPLING FOR CLUSTERING AND ANOMALY DETECTION

Clustering is to segment similar records into the same group according to certain characteristics, and those records that cannot be classified into any group may be abnormal points. The challenge that clustering technology encounters in the era of big data is also the problem of data volume, and the clustering operation itself consumes a lot of calculations. Shirkhorshidi *et al.* [83] divide big data clustering into two categories: single-machine clustering and multiple-machine clustering. Single column reduces the amount of data by using data reduction methods, e.g., sampling and dimensionality reduction. Multi-machine clustering refers to the use of parallel distributed computing frameworks, e.g., MapReduce and cluster resources to increase computing power.

Kollios *et al.* [59] propose biased sampling to speed up clustering and anomaly detection on big data. Unlike the previous work, they consider the data characteristics and analysis goals during the sampling process. Based on the tasks of clustering and anomaly detection, Kollios *et al.* [59] consider the data density problem in the dataset. They propose a biased sampling method to improve the accuracy of clustering and anomaly detection. The biased sampling is to make the data points in each cluster and the abnormal points have a higher probability of being selected. In order to achieve this goal, they use the density estimation method to estimate density around the data points. In the experiment, they prove that density-based sampling has a better effect on clustering than uniform sampling.

Figure 6 shows the use of biased samples in clustering. Figure 6(a) is the distribution of the original data and there are three classes with higher density. Figure 6(b) is the result of random sampling on the original data set. Figure 6(c) is the result of applying the biased sampling to the original data. Figure 6(d) shows 10 data points selected from each of the three categories clustered based on the random sampling, and Figure 6(e) shows 10 data points selected from each of the three categories clustered based on the biased sampling method. After comparison with the categories in the original data, it is found that the clustering results of the biased samples are more accurate.

C. SAMPLING FOR SUMMARIES AND SKETCHES

Summaries or sketches can be performed by sampling or hashing data values to a smaller domain [12]. Although different scholars have applied different sampling algorithms, the most commonly used sampling algorithm among data

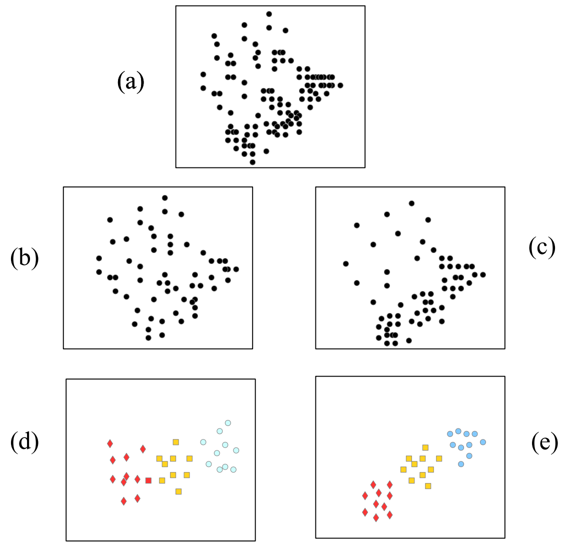


FIGURE 6. Application of biased sampling in clustering tasks [59].

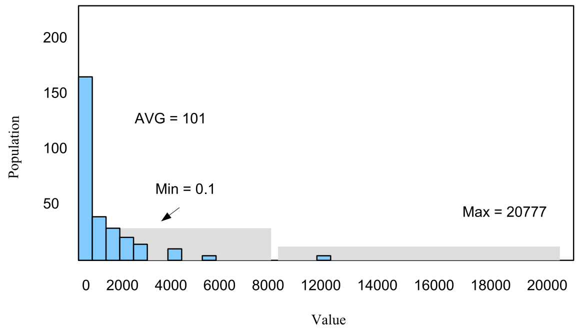


FIGURE 7. Sparseness of one representative production data [84].

scientists is random sampling [61]. The main reason is that random sampling is the best and easiest to use, which is the only technique commonly used by data scientists to quickly gain insights from big data sets.

Rojas *et al.* [61] first interview 22 data scientists working on large data sets and find that they basically use random sampling or pseudo-random sampling. Certainly, these data scientists believe that other sampling techniques may achieve better results than random sampling. These scientists perform a data exploration task that used different sampling methods to support classification of more than 2 million generated samples from data records of Wikipedia article edit. Research has shown that sampling techniques other than random sampling can generate insights into the data, which can help focus on the different characteristics of the data without affecting the quality of data exploration and helping people understand the data. This shows that with the application of sampling, Summaries or sketches of data can be created to help scientist observe and understand the data.

Aggregated queries are also a way to generate summaries of data. Aggregate queries are computationally expensive which need to traverse the data. In the era of big data, a single machine often cannot make such a large amount of data. Therefore, aggregate queries for big data are often performed on distributed systems that scales to thousands of machines. The commonly used distributed computing frameworks are Hadoop, spark, etc. Although distributed systems provide tremendous parallelism to improve performance, the processing cost of aggregated queries remains high [60]. Investigation in one cluster of [60] reveals that 90% of 2,000 data mining jobs are aggregation queries. These queries consume two-thousand machine hours on average, and some of them take up to 10 hours.

Therefore, Yan *et al.* [60] use sampling technique to reduce the amount of data. When error bounds cannot be compromised and data is sparse, they think that conventional uniform sampling often yields high sampling rates and thus deliver limited or no performance gains. For example, uniform sampling with 20% error bound and 95% confidence needs to consume 99.91% of the data whose distribution is shown in Figure 7. Hence they propose error-bounded stratified sampling, which is a variant of stratified sampling [84] and relies on the insight, i.e., prior knowledge of data distribution, to reduce sample size. Error bound means that the real value has a large probability of falling within an interval. Sparse data means that the data is generally limited but wide-ranging.

Taking the data distribution in Figure 7 as an example, error-bounded stratified sampling can divide the data into two groups. One group covers the header data and the other covers the tail data. Because the data range of the first group is small, the sampling rate is also small. Although the data range of the second group is large, the data basically falls in the first group. Even if the data of the second group is all taken as a sample, the overall sampling rate is still low. It is worth mentioning that the technique has been implemented into Microsoft internal search query platform.

D. SAMPLING FOR REGRESSION ANALYSIS

Statistical analysis such as regression analysis can be used to analyze the relationship between multiple columns in a relation. Sauter [85] think that statistics are learned from data. Statistics methods are often used for data profiling, which have encountered the problem of excessive data volume in the era of big data. Statistical analysis of the entire big data set requires a certain amount of calculation and time.

Under the computational pressure of large data sets, many traditional statistical methods are no longer applicable. Although sampling can help with data reduction, how to avoid sampling errors caused by sampling needs to be considered. For example, [62] mention that in the context of linear regression, traditional sub-sampling methods are prone to introduce sampling errors and affect the covariance matrix of the estimator. Hence, they propose information-based optimal subdata selection method called IBOSS. The goal of IBOSS is to select data points that are informative so that small-sized subdata retains most of the information contained in the complete data. Simulation experiments prove that IBOSS is faster and suitable for distributed parallel computing.

Jun *et al.* [63] propose to use sampling to divide big data into some sub data sets in regression problem for reducing the computing burden. The traditional statistical analysis of big data is to sample from big data, and then perform statistical analysis on the sample to infer the population. Jun *et al.* [63] divide the big data closed to population into some sub data sets with small size closed to sample which is proper for big data analysis. They treat the entire data set as a population and the sub set as a sample to reduce computing burden. And they select regression analysis to perform experiments. The traditional processing is shown in Figure 8, and their design is shown in Figure 9.

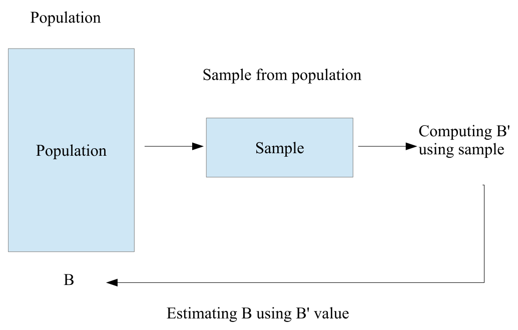


FIGURE 8. Traditional big data regression analysis [63].

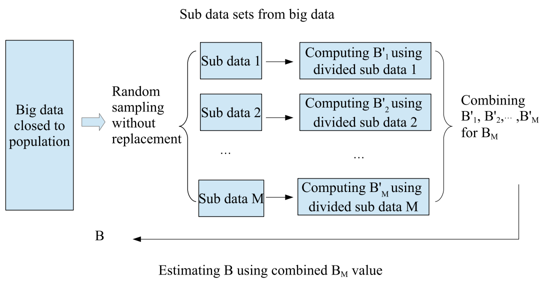

Sub data sets from big data
Big data closed to population
Random sampling without replacement
Sub data 1
Sub data 2
...
Sub data M
Computing B'1 using divided sub data 1
Computing B'2 using divided sub data 2
...
Computing B'M using divided sub data M
Combining B'1, B'2,··· ,B'M for BM
B
Estimating B using combined BM value

FIGURE 9. Sampling-based partitioning big data regression analysis [63].

Their design consists of three steps: the first step is to first generate M sub-data sets using random samples without replacement; the second step is to calculate the regression parameters of each sub-data set and calculate the average of regression parameters of the M sub-data sets; the third step is to use the averaged parameters obtained in the second step to estimate regression parameters on the entire data set. This design that combines sampling and parallel processing helps them speed up regression analysis on big data. By experimenting with the data set from the simulation and UCI machine learning repository, the author proves that the regression parameters obtained by distributed calculation on random samples are close to the regression parameters calculated on entire data set. This provides a reference for statistical analysis on the entire large data set.

V. SAMPLING FOR DEPENDENCIES

A dependency is a metadata that describes the relationship between columns in relation, based on either value equality or similarity [86]. There are many use cases for dependencies. For example, unique column combinations are used for finding key attributes in relation [64], and functional dependencies can be used for schema normalization [88] or consistent query answering [89], while inclusion dependencies can suggest how to join two relations [12]. Inclusion dependencies together with functional dependencies form the most important data dependencies used in practice [87]. But discovery of dependencies is time consuming and memory consuming. Many functional dependencies discovery algorithms are not suitable for large data sets. Sampling could be employed to estimate the support and confidence measures of data dependencies [91], [92]. By sampling, you can select a small enough representative data set from the big data set. Hence the choice of sampling method is very important, which help to ensure that the estimated inaccuracy rate is below a predefined bound with high confidence. Specifically, based on the classification for dependency in [12], we investigate sampling for unique column combinations in Section V-A, functional dependency in Section V-B, inclusion dependency in Section V-C.

A. SAMPLING FOR DISCOVERY OF UNIQUE COLUMN COMBINATIONS

An important goal in data profiling is to find the right key for the relational table, e.g., primary key. The step before key discovery is to discover unique column combinations. Unique column combinations are sets of columns whose values uniquely identify rows, which is an important data profiling task [93]. But discovery of unique column combinations is computationally expensive, which is suitable for small dataset or samples of large dataset. For large data set, sampling is a promising method for knowledge discovery [94]. Based on sampling-based knowledge discovery, it is necessary to first select samples from the entire data set and obtain knowledge from the samples, and then use the entire data set to verify that the acquired knowledge is correct.

A typical algorithm for identifying key attributes is GORDIAN proposed by Sismanis *et al.* [64]. The main idea of GORDIAN is to turn the problem of keys identification into cube computation problem, and then find non-keys through cube computation. Finally, GORDIAN calculates the complement of the non-keys set to obtain the desired set of keys. Therefore, the GORDIAN algorithm can be divided into three steps: (*i*) create the prefix tree through a single pass over the data; (*ii*) find maximal non-uniques by traversing the prefix tree with prunning; (*iii*) get minimal keys from set of maximal non-uniques. In order to make GORDIAN scalable to large datasets, Sismanis *et al.* combine GORDIAN with sampling. Experiments have shown that sampling-based GORDIAN can find all true keys and approximate keys using only a relatively small number of samples.

GORDIAN algorithm is further developed by Abedjan and Naumann [65] to discover unique column combinations. Since the existing algorithms are either too violent or have high memory requirements and cannot be applied to big data sets. A hybrid solution HCA-Gordian, which combines Gordian algorithm [64] and their new algorithm the Histogram-Count-based Apriori Algorithm (HCA), is proposed by Abedjan and Naumann [65] to discover unique column combinations. GORDIAN algorithm is used to find composite keys and the HCA is an optimized bottom-up algorithm which takes efficient candidate generation and statistics-based pruning methods. HCA-Gordian performs Gordian algorithm on a smaller sample of table to discover non-uniques and non-uniques will be used as pruning candidates when executing HCA on the entire table.

In the experiment setup, the sample size for the preprocessing step of non-unique discover is always 10,000 tuple sample. Especially when the amount of data is large and the number of unique is small, the runtime of HCA-Gordian is lower than Gordian. For example, when using real world tables for experiments, search speed of HCA-Gordian is four times faster than Gordian. And as the data set grows larger, e.g., the National file contains 1,394,725 tuples, Gordian takes too long to run, while HCA-Gordian only takes 115 seconds to complete. In addition, When the number of detected non-uniques is high, the discovery effect of HCA-Gordian is better than Gordian.

B. SAMPLING FOR FUNCTIONAL DEPENDENCIES

A functional dependency refers to a set of attributes in a relationship that determines another set of attributes. For example, there is such a functional dependency A->B, which means that any two records in the relationship, when their values on the attribute set A are equal, the values on the attribute set B must be equal. Bleifuß *et al.* [66] propose an approximate discovery strategy AID-FD (Approximate Iterative Discovery of FDs) which sacrifices a certain correct rate in exchange for performance improvement. AID-FD uses an incremental, focused sampling of tuple pairs to deduce non-FDs until user-configured termination criterion is met. The authors have demonstrated in experiments that the AID-FD method uses only 2%-40% of the time of the exact algorithm when processing the same data set, but finds more than 99% of the functional dependencies.

Papenbrock and Naumann [67] mention that today's various functional dependencies discovery algorithms do not have the ability to process more than 50 columns and 1 million rows of data. Thus, they propose the sampling-based FD discovery algorithm HYFD. And there are three properties in sampling-based FD discovery algorithms: Completeness, Minimality, Proximity, which are important for HYFD. HYFD combines column-efficient FD induction techniques with row-efficient FD search techniques in two phases. In Phase 1, they apply focused sampling techniques to select samples with a possibly large impact on the result's precision and produce a set of FD candidates based on samples. In Phase 2, the algorithm applies row-efficient FD search techniques to validate the FD candidates produced in Phase 1. The sampling method allows functional dependencies discovery algorithms to be extended to large data sets.

In experiments, when the data set is not very large, the runtime of HYFD is almost all lower than other algorithms. When the data set exceeds 50 columns and 10 million rows, HYFD can get the result through a few days of calculation. However, other algorithms cannot complete the calculation, because the time complexity for these algorithms is exponential. This again demonstrates that sampling is important for data profiling, e.g., FD discovery.

In the above, we mention that using focused sampling to find functional dependencies. In this section, we will mention the use of random sampling to find soft functional dependencies. The so-called ‘‘soft’’ functional dependency is relative to the ‘‘hard’’ functional dependency. A ‘‘hard’’ functional dependency means that the entire relationship satisfies the functional dependency, while a ‘‘soft’’ functional dependency means that the entire relationship is almost satisfied, or that there is a high probability of satisfying the functional dependency.

Ilyas *et al.* [68] propose sampling-based CORDS, which means that automatic discovery of correlations and soft functional dependencies between columns, to find approximate dependencies. Among them, correlation refers to the general statistical dependence, while soft functional dependence refers to that value of attribute C1 determines the value of attribute C2 with high probability. CORDS use enumeration to generate pairs of columns that may be associated, and heuristically cuts out those unrelated column pairs with high probable. CORDS apply random sampling with replacement to generate sample. In the implementation of CORDS, they only use a few hundred rows of sample data, and the sample size is independent of the data size. In the experiment to evaluate the advantages of applying CORDS, where run a workload of 300 queries on the Accidents database, the median query execution time and worst query execution time with CORDS applied were better than those without CORDS. Hence CORDS is efficient and scalable when it encounters large-scale dataset.

Approximate functional dependence is similar to the meaning of soft functional dependency. Approximate functional dependence requires the normal functional dependency to be satisfied by most tuples of relation R [95], [96]. Of course, approximation functional dependencies contain exact functional dependencies that are satisfied throughout the relationship. As mentioned in [95], when the amount of data is large, the time for discovery of functional dependency will increase exponentially. Therefore, Kivinen and Mannila [95] propose to discover approximate dependencies by random sampling. In fact, sampling can be used not only to find approximate functional dependencies, but also to verify exact functional dependencies [96]. If the exact functional dependency does not satisfy all the sample data, then the whole relationship is definitely not satisfied, hence such functional dependencies can be removed.

Functional dependencies are satisfied for all tuples in the relation, while conditional functional dependencies (CFDs) is to hold on the subset of tuples that satisfies some patterns [97]. And CFDs can be used for data cleaning [97], [98]. Fan *et al.* [90] propose three methods for conditional functional dependencies discovery. However, when the size of data set is large, no dependency discovery algorithms scale very well to discover minimal conditional functional dependencies.

When mining CFDs on big data, the volume issue of big data has to be solved. Li *et al.* [69] develop the sampling algorithms to obtain a small representative training set from large and low-quality datasets and discover CFDs on the samples. They use sampling technology for two reasons. One is that finding CFD needs to scan the data set multiple times, and sampling helps reduce the amount of data. The second is to use the sampling method to help them filter those dirty items on the low-quality data set and choose clean items as the training set. They define criteria for misleading tuples, which are dirty, incomplete or very similar to popular tuples. And then they design a Representative and Random Sampling for CFDs (BRRSC), which is similar to reservoir sampling [39]. The difference is that they combine the criteria defined above during the sampling process. Furthermore, they propose fault-tolerant CFDs discovery and conflict-resolution algorithms to find CFDs. Finally, experimental results show that their sampling-based CFD discovery algorithms can find valid CFD rules for billions of data in a reasonable time.

C. SAMPLING-BASED TEST FOR INCLUSION DEPENDENCY CANDIDATES

The definition of inclusion dependencies (INDs) is that the combination of values that appear in a set of attribute columns must also appear in another set of attribute columns [99]. Therefore, inclusion dependencies are often used to discover foreign keys [87]. However, discovery of inclusion dependencies is computationally expensive. One of the reasons is that the existing algorithms need to shuffle huge amounts of data to test inclusion dependencies candidates, which puts pressure on both computing and memory [70].

Under these circumstances, Kruse *et al.* [70] propose fast approximate discovery of inclusion dependencies (FAIDA). FAIDA can guarantee to find all INDs and only false positives with a low probability in order to balance efficiency and correctness. FAIDA uses algorithms [100], [101] of Apriori-style to generate inclusion dependencies candidates. The inverted index values and operates on a small sample of the input data. The sampling algorithm is applied to each table to get each sample. Rather than use random sampling to get sample, they assure that sample table contains min $\{s, d_A\}$ distinct values for each column A, where s represents sample size and d_A represents number of distinct values in column A.

In their experiments, they set sample size to a default of 500. In order to verify the efficiency of FAIDA, Kruse *et al.* [70] compare FAIDA's runtime with the state-of-the-art algorithm for exact IND discovery BINDER [101] on multiple datasets. On four datasets, FAIDA is steadily 5 to 6 times faster than BINDER, and they generate and test almost the same number of IND candidates. Especially when one of the datasets reaches 79.4GB, BINDER takes 9 hours and 32 minutes to complete, while FAIDA only takes 1 hour and 47 minutes. Their evaluation shows that sampling-based FAIDA outperforms the state-of-the-art algorithm by a factor of up to six in terms of runtime without reporting any false positives.

VI. SUMMARY AND FUTURE WORKS

Data in various fields are increasing on a large scale. Big data brings us new opportunities and challenges. Through data analysis and data mining of big data, we can get a lot of potential value. However, due to the large amount of data, it brings great challenges to the processing and storage. Therefore, data analysis, data mining or data profiling on large data sets have to face the pressure of calculation and time. Increasing computing power by using clusters of computers is one solution, but many times this is not the case, and designing distributed computing is often difficult. Hence the application of data reduction techniques like sampling is very important. There are some mature research articles on data profiling and sampling, but ''sampling for big data profiling'' does not exist, therefore this article focuses on researching sampling for data profiling tasks in big data context. This paper introduces the application of sampling in data profiling tasks. According to the classification of data profiling in [12], we introduce the application of sampling in single column data profiling, multiple columns data profiling and dependency discovery. In conclusion, Table 3 summarizes the sampling for data profiling tasks investigated in survey, indicating the widespread use of sampling in data profiling.

The above survey on ''sampling for big data profiling'' is mainly about relational databases, and rarely involves graph data or time series data. Since there is less research on sampling-based data profiling for graph data or time series data, we provide some future directions as follows.

A. SAMPLING FOR PROFILING TIME SERIES DATA

Many tasks on time series data need data profiling, e.g., matching heterogeneous events in a sequence [102], repairing timestamps according to the given temporal constraints [103] such as sequential dependencies [104]. All these studies use data profiling to detect and repair erroneous temporal data. The computational cost and time cost in large-scale temporal data streams can be high. Therefore, sampling for profiling time series data is valuable and necessary.

In the time series data stream, we do not need to get exact results, e.g., when calculating the quantiles or probability distributions of speeds. Approximate results are valuable in time-series data streams, for example approximate probability distributions of speeds can also help us perform effective anomaly detection. In the sampling of time series data, statistical probability distributions of speeds are different from discovering quantiles. The speed of time series data depends on the adjacent time-series data points, which means that sampling for calculating speed of time series requires a set of data points in a window. Therefore, how to apply the sampling technology to the aforesaid data profiling task of time series data needs further experimental analysis and research.

B. SAMPLING FOR PROFILING GRAPH DATA

Data profiling is also heavily used in graph data, e.g., using Petri Nets in process mining to recover missing events [105], [106] and clean event data [107], discovering keys for graphs and applying keys to study entity matching [108], or defining functional dependencies for graphs [21] and discovering them [109]. However, the above studies still seem to be difficult when encountering large graphs. Fan *et al.* [108] prove that entity matching is NP-complete for graphs and recursively defined keys for graphs bring more challenges. In this case, one has to design two parallel scalable algorithms, in MapReduce and a vertex-centric asynchronous model. In order to find Graph Functional Dependencies, Fan *et al.* [109] have to deal with large-scale graphs by designing effective pruning strategies, using parallel algorithms, and adding processors. As mentioned earlier, designing parallel algorithms is difficult.

Equivalently, profiling for graph data has to face the pressure of computing and memory when data profiling encounters large graphs. Therefore, it is necessary and worth researching to sample the graph data and carry out the tasks of data profiling based on the sample. But sampling graph data is more difficult than sampling relational data. Leskovec and Faloutsos [37] did practical experiments on sampling from large graphs. They concluded that best performing methods are the ones based on random-walks and ''forest fire'', with sample sizes as low as 15% of the original graph. However, how to apply these graph sampling methods to the above-mentioned graph data-based data profiling tasks is waiting for further experiments and exploration.

C. SAMPLING FOR PROFILING HETEROGENEOUS DATA

Data profiling is also widely used for heterogeneous data, e.g., discovering matching dependencies (MDs) [110], [111], reasoning about matching rules [112], [113], discovering a concise set of matching keys [114] and conditional matching dependencies (CMDs) [115]. However, these profiling tasks also have to face computational pressure in a big data context.

In fact, MDs, DDs and data dependencies are all based on differential functions. When calculating the measures for differential dependencies, performing sampling of pairwise comparison is more difficult. Given an instance of relation R with N data tuples, pairwise comparison M will increase the total number to $\frac{N*(N-1)}{2}$, which will greatly increase the number of populations. However, many pairs in M are meaningless when calculating support for DDs [116], which means that the proportion of pairs we want is very small. Therefore, we must increase the sampling rate to expect to include these pairs in the sample, so as to get the approximate results as close as possible.

REFERENCES

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, Apr. 2014.
[2] J. Manyika. (2011). *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. [Online]. Available: http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation
[3] Ishwarappa and J. Anuradha, "A brief introduction on big data 5 Vs characteristics and Hadoop technology," *Procedia Comput. Sci.*, vol. 48, pp. 319–324, Jan. 2015.
[4] B. Chen, P. Haas, and P. Scheuermann, "A new two-phase sampling based algorithm for discovering association rules," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining - KDD*, 2002, pp. 462–468.
[5] A. Bifet, "Mining big data in real time," *Informatica*, vol. 37, no. 1, pp. 15–20, 2013.
[6] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.
[7] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining Anal.*, vol. 3, no. 2, pp. 85–101, Jun. 2020.
[8] S. González-Bailón, "Social science in the era of big data," *Policy Internet*, vol. 5, no. 2, pp. 147–160, Jun. 2013.
[9] W. Albattah, "The role of sampling in big data analysis," in *Proc. Int. Conf. Big Data Adv. Wireless Technol. BDAW*, Blagoevgrad, Bulgaria, 2016, p. 28.
[10] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara, "Evaluation of sampling for data mining of association rules," in *Proc. 7th Int. Workshop Res. Issues Data Eng. High Perform. Database Manage. Large-Scale Appl.*, Birmingham, U.K., 1997, pp. 42–50.
[11] D. Pyle, *Data Preparation for Data Mining*. San Mateo, CA, USA: Morgan Kaufmann, 1999.
[12] Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: A survey," *VLDB J.*, vol. 24, no. 4, pp. 557–581, Aug. 2015.
[13] S. Song, A. Zhang, L. Chen, and J. Wang, "Enriching data imputation with extensive similarity neighbors," *Proc. VLDB Endowment*, vol. 8, no. 11, pp. 1286–1297, Jul. 2015.
[14] S. Song, H. Zhu, and J. Wang, "Constraint-variance tolerant data repairing," in *Proc. Int. Conf. Manage. Data SIGMOD*, San Francisco, CA, USA, Jun. 2016, pp. 877–892.
[15] A. S. Singh and M. B. Masuku, "Sampling techniques & determination of sample size in applied statistics research: An overview," *Int. J. Econ., Commerce Manage.*, vol. 2, no. 11, pp. 1–22, 2014.
[16] F. Naumann, "Data profiling revisited," *ACM SIGMOD Rec.*, vol. 42, no. 4, pp. 40–49, Feb. 2014.
[17] Z. Abedjan, L. Golab, and F. Naumann, "Data profiling," in *Proc. 32nd IEEE Int. Conf. Data Eng., ICDE*, Helsinki, Finland, May 2016, pp. 1432–1435.
[18] S. Song, A. Zhang, J. Wang, and P. S. Yu, "SCREEN: Stream data cleaning under speed constraints," in *Proc. ACM SIGMOD Int. Conf. Manage. Data SIGMOD*, Victoria, Australia, May 2015, pp. 827–841.
[19] A. Zhang, S. Song, and J. Wang, "Sequential data cleaning: A statistical approach," in *Proc. Int. Conf. Manage. Data SIGMOD*, San Francisco, CA, USA, 2016, pp. 909–924.
[20] F. Wang, M. Li, Y. Mei, and W. Li, "Time series data mining: A case study with big data analytics approach," *IEEE Access*, vol. 8, pp. 14322–14328, 2020.
[21] W. Fan, Y. Wu, and J. Xu, "Functional dependencies for graphs," in *Proc. Int. Conf. Manage. Data - SIGMOD*, San Francisco, CA, USA, 2016, pp. 1843–1857.
[22] S. Song, H. Cheng, J. X. Yu, and L. Chen, "Repairing vertex labels under neighborhood constraints," *Proc. VLDB Endowment*, vol. 7, no. 11, pp. 987–998, Jul. 2014.
[23] S. Song, B. Liu, H. Cheng, J. X. Yu, and L. Chen, "Graph repairing under neighborhood constraints," *VLDB J.*, vol. 26, no. 5, pp. 611–635, Oct. 2017.
[24] D. Maier, Y. Alon Halevy, and J. Michael Franklin, "Dataspaces: Co-existence with heterogeneity," in *Proc. 10th Int. Conf. Princ. Knowl. Represent. Reasoning, Lake District United Kingdom*, Jun. 2006, p. 3.

[25] S. Song, L. Chen, and P. S. Yu, "On data dependencies in dataspaces," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Hannover, Germany, Apr. 2011, pp. 470–481.

[26] S. Song, L. Chen, and P. S. Yu, "Comparable dependencies over heterogeneous data," *VLDB J.*, vol. 22, no. 2, pp. 253–274, Apr. 2013.

[27] Z. Abedjan, L. Golab, and F. Naumann, "Data profiling: A tutorial," in *Proc. ACM Int. Conf. Manage. Data SIGMOD*, Chicago, IL, USA, 2017, pp. 1747–1751.

[28] H. T. Schreuder, T. G. Gregoire, and J. P. Weyer, "For what applications can probability and non-probability sampling be used?" *Environ. Monitor. Assessment*, vol. 66, no. 3, pp. 281–291, 2001.

[29] R. Iachan, L. Berman, T. M. Kyle, K. J. Martin, Y. Deng, D. N. Moyse, D. Middleton, and A. A. Atienza, "Weighting nonprobability and probability sample surveys in describing cancer catchment areas," *Cancer Epidemiology, Biomarkers Prevention, A Publication Amer. Assoc. Cancer Res., Cosponsored Amer. Soc. Preventive Oncol.*, vol. 28, no. 3, pp. 471–477, Mar. 2019.

[30] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (Statistical) learning tools for our era of data deluge," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 18–31, Sep. 2014.

[31] R. Agrawal, A. Kadadi, X. Dai, and F. Andrès, "Challenges and opportunities with big data visualization," in *Proc. 7th Int. Conf. Manage. Comput. Collective Intell. Digit. Ecosyst.*, Caraguatatuba, Brazil, Oct. 2015, pp. 169–173.

[32] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, May 2017.

[33] C. Kadilar and H. Cingi, "Ratio estimators in simple random sampling," *Appl. Math. Comput.*, vol. 151, no. 3, pp. 893–902, 2004.

[34] P. J. Bickel and D. A. Freedman, "Asymptotic normality and the bootstrap in stratified sampling," *Ann. Statist.*, vol. 12, no. 2, pp. 470–482, Jun. 1984.

[35] H. J. G. Gundersen and E. B. Jensen, "The efficiency of systematic sampling in stereology and its prediction," *J. Microsc.*, vol. 147, no. 3, pp. 229–263, Sep. 1987.

[36] R. H. Henderson and T. Sundaresan, "Cluster sampling to assess immunization coverage: A review of experience with a simplified sampling method," *Bull. World Health Org.*, vol. 60, no. 2, p. 253, 1982.

[37] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD*, Philadelphia, PA, USA, 2006, pp. 631–636.

[38] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin, and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Proc. 1st Int. Conf. Adv. Data Inf. Eng., DaEng.*, Kuala Lumpur, Malaysia, Dec. 2013, pp. 13–22.

[39] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, Mar. 1985.

[40] Q. He, H. Wang, F. Zhuang, T. Shang, and Z. Shi, "Parallel sampling from big data with uncertainty distribution," *Fuzzy Sets Syst.*, vol. 258, pp. 117–133, Jan. 2015.

[41] Y. He, J. Z. Huang, H. Long, Q. Wang, and C. Wei, "I-sampling: A new block-based sampling method for large-scale dataset," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Honolulu, HI, USA, Jun. 2017, pp. 360–367.

[42] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: A survey," *J. Big Data*, vol. 2, no. 1, p. 21, 2015.

[43] H. George John and P. Langley, "Static versus dynamic sampling for data mining," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, Portland, OR, USA, 1996, pp. 367–370.

[44] F. Provost, D. Jensen, and T. Oates, "Efficient progressive sampling," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD*, San Diego, CA, USA, 1999, pp. 23–32.

[45] T. Harford, "Big data: A big mistake?" *Significance*, vol. 11, no. 5, pp. 14–19, Dec. 2014.

[46] J. Nagler and J. A. Tucker, "Drawing inferences and testing theories with big data," *PS, Political Sci. Politics*, vol. 48, no. 1, pp. 84–88, Jan. 2015.

[47] J. K. Kim and Z. Wang, "Sampling techniques for big data analysis," *Int. Stat. Rev.*, vol. 87, no. S1, pp. S177–S191, May 2019.

[48] S.-M. Tam and J.-K. Kim, "Big data ethics and selection-bias: An official statistician's perspective," *Stat. J. IAOS*, vol. 34, no. 4, pp. 577–588, Dec. 2018.

[49] R. Maisel, "Inference from nonprobability samples," *Public Opinion Quart.*, vol. 36, no. 3, p. 407, Sep. 1972.

[50] X.-L. Meng, "Statistical paradises and paradoxes in big data (I): Law of large populations, big data paradox, and the 2016 US presidential election," *Ann. Appl. Statist.*, vol. 12, no. 2, pp. 685–726, Jun. 2018.

[51] J. Peter Haas, F. Jeffrey Naughton, S. Seshadri, and L. Stokes, "Sampling-based estimation of the number of distinct values of an attribute," in *Proc. 21th Int. Conf. Very Large Data Bases VLDB*, Zurich, Switzerland, Sep. 1995, pp. 311–322.

[52] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya, "Towards estimation error guarantees for distinct values," in *Proc. 19th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst. PODS*, Dallas, TX, USA, 2000, pp. 268–279.

[53] B. Phillip Gibbons, "Distinct sampling for highly-accurate answers to distinct values queries and event reports," in *Proc. 27th Int. Conf. Very Large Data Bases VLDB*, Roma, Italy, Sep. 2001, pp. 541–550.

[54] S. Chaudhuri, R. Motwani, and R. V. Narasayya, "Random sampling for histogram construction: How much is enough?" in *Proc. ACM SIGMOD Int. Conf. Manage. Data SIGMOD*, Washington, DC, USA, Jun. 1998, pp. 436–447.

[55] P. B. Gibbons, Y. Matias, and V. Poosala, "Fast incremental maintenance of approximate histograms," *ACM Trans. Database Syst.*, vol. 27, no. 3, pp. 261–298, Sep. 2002.

[56] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, "Random sampling techniques for space efficient online computation of order statistics of large datasets," in *Proc. ACM Int. Conf. Manage. Data SIGMOD*, Philadelphia, PA, USA, Jun. 1999, pp. 251–262.

[57] Z. Huang, L. Wang, K. Yi, and Y. Liu, "Sampling based algorithms for quantile computation in sensor networks," in *Proc. Int. Conf. Manage. Data SIGMOD*, Athens, Greece, 2011, pp. 745–756.

[58] X. Wu, W. Fan, J. Peng, K. Zhang, and Y. Yu, "Iterative sampling based frequent itemset mining for big data," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 6, pp. 875–882, Dec. 2015.

[59] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 5, pp. 1170–1187, Sep. 2003.

[60] Y. Yan, L. J. Chen, and Z. Zhang, "Error-bounded sampling for analytics on big sparse data," *Proc. VLDB Endowment*, vol. 7, no. 13, pp. 1508–1519, Aug. 2014.

[61] J. A. Ramos Rojas, M. Beth Kery, S. Rosenthal, and A. Dey, "Sampling techniques to improve big data exploration," in *Proc. IEEE 7th Symp. Large Data Anal. Visualizat. (LDAV)*, Phoenix, AZ, USA, Oct. 2017, pp. 26–35.

[62] H. Wang, M. Yang, and J. Stufken, "Information-based optimal subdata selection for big data linear regression," *J. Amer. Stat. Assoc.*, vol. 114, no. 525, pp. 393–405, Jan. 2019.

[63] S. Jun, S.-J. Lee, and J.-B. Ryu, "A divided regression analysis for big data," *Int. J. Softw. Eng. Appl.*, vol. 9, no. 5, pp. 21–32, May 2015.

[64] Y. Sismanis, P. Brown, J. P. Haas, and B. Reinwald, "GORDIAN: Efficient and scalable discovery of composite keys," in *Proc. 32nd Int. Conf. Very Large Data Bases*, Seoul South Korea, Sep. 2006, pp. 691–702.

[65] Z. Abedjan and F. Naumann, "Advancing the discovery of unique column combinations," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. - CIKM*, Glasgow, U.K., 2011, pp. 1565–1570.

[66] T. Bleifuß, S. Bülow, J. Frohnhofen, J. Risch, G. Wiese, S. Kruse, T. Papenbrock, and F. Naumann, "Approximate discovery of functional dependencies for large datasets," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage. CIKM*, Indianapolis, IN, USA, 2016, pp. 1803–1812.

[67] T. Papenbrock and F. Naumann, "A hybrid approach to functional dependency discovery," in *Proc. Int. Conf. Manage. Data SIGMOD*, San Francisco, CA, USA, 2016, pp. 821–833.

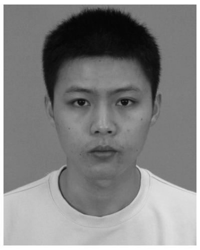
[68] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga, "CORDS: Automatic discovery of correlations and soft functional dependencies," in *Proc. ACM SIGMOD Int. Conf. Manage. Data SIGMOD*, Paris, France, 2004, pp. 647–658.

[69] M. Li, H. Wang, and J. Li, "Mining conditional functional dependency rules on big data," *Big Data Mining Anal.*, vol. 3, no. 1, pp. 68–84, Mar. 2020.

[70] S. Kruse, T. Papenbrock, C. Dullweber, M. Finke, M. Hegner, M. Zabel, C. Zöllner, and F. Naumann, "Fast approximate discovery of inclusion dependencies," in *Datenbanksysteme Für Business, Technologie Und Web BTW*, Stuttgart, Germany, 2017, pp. 207–226.

[71] R. Huang, Z. Chen, Z. Liu, S. Song, and J. Wang, "TsOutlier: Explaining outliers with uniform profiles over IoT data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 2024–2027.

[72] Z. Abedjan, L. Golab, F. Naumann, and T. Papenbrock, "Data Profiling," in *Synthesis Lectures on Data Management*. San Rafael, CA, USA: Morgan & Claypool, 2018.
[73] S. Song, H. Zhu, and L. Chen, "Probabilistic correlation-based similarity measure on text records," *Inf. Sci.*, vol. 289, pp. 8–24, Dec. 2014.
[74] H. Harmouch and F. Naumann, "Cardinality estimation: An experimental survey," *Proc. VLDB Endowment*, vol. 11, no. 4, pp. 499–512, Dec. 2017.
[75] R. Kooi, "The Optim. Queries relational databases," Ph.D. dissertation, Dept. Comput. Eng. Sci., Case Western Reserve Univ., Cleveland, OH, USA, Sep. 1980.
[76] G. P. Selinger, M. M. Astrahan, D. D. Chamberlin, A. R. Lorie, and G. T. Price, "Access path selection in a relational database management system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Boston, MA, USA, May 1979, pp. 23–34.
[77] I. Pohl, *A Minimum Storage Algorithm for Computing the Median*. Ossining, NY, USA: IBM TJ Watson Research Center, 1969.
[78] L. Zhang, W. Wang, and Y. Zhang, "Privacy preserving association rule mining: Taxonomy, techniques, and metrics," *IEEE Access*, vol. 7, pp. 45032–45047, 2019.
[79] S. Song, C. Li, and X. Zhang, "Turn waste into wealth: On simultaneous clustering and cleaning over dirty data," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD*, Sydney, NSW, Australia, 2015, pp. 1115–1124.
[80] A. Zhang, S. Song, Y. Sun, and J. Wang, "Learning individual models for imputation," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 160–171.
[81] R. Agrawal and C. J. Shafer, "Parallel mining of association rules," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 962–969, Dec. 1996.
[82] D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," in *Proc. 4th Int. Conf. Parallel Distrib. Inf. Syst.*, Miami Beach, FL, USA, 1996, pp. 31–42.
[83] A. S. Shirkhorshidi, S. R. Aghabozorgi, Y. W. Teh, and T. Herawan, "Big data clustering: A review," in *Proc. Int. Conf. Comput. Sci. Appl.*, Guimaraes, Portugal, Jun. 2014, pp. 707–720.
[84] P. M. D. Explained, "Sampling: Design and analysis," *Technometrics*, vol. 42, no. 2, pp. 223–224, 2009.
[85] R. Sauter, "Introduction to probability and statistics for engineers and scientists," *Technometrics*, vol. 47, no. 3, p. 378, 2005.
[86] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM Trans. Database Syst.*, vol. 36, no. 3, pp. 1–41, Aug. 2011.
[87] S. Lopes, J.-M. Petit, and F. Toumani, "Discovering interesting inclusion dependencies: Application to logical database tuning," *Inf. Syst.*, vol. 27, no. 1, pp. 1–19, Mar. 2002.
[88] T. Papenbrock and F. Naumann, "Data-driven schema normalization," in *Proc. 20th Int. Conf. Extending Database Technol. EDBT*, Venice, Italy, Mar. 2017, pp. 342–353.
[89] X. Lian, L. Chen, and S. Song, "Consistent query answers in inconsistent probabilistic databases," in *Proc. Int. Conf. Manage. Data SIGMOD*, Indianapolis, IN, USA, 2010, pp. 303–314.
[90] W. Fan, F. Geerts, J. Li, and M. Xiong, "Discovering conditional functional dependencies," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 5, pp. 683–698, May 2011.
[91] S. Song, L. Chen, and H. Cheng, "Parameter-free determination of distance thresholds for metric distance constraints," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Washington, DC, USA, Apr. 2012, pp. 846–857.
[92] S. Song, L. Chen, and H. Cheng, "Efficient determination of distance thresholds for differential dependencies," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2179–2192, Sep. 2014.
[93] A. Heise, J.-A. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann, "Scalable discovery of unique column combinations," *Proc. VLDB Endowment*, vol. 7, no. 4, pp. 301–312, Dec. 2013.
[94] J. Kivinen and H. Mannila, "The power of sampling in knowledge discovery," in *Proc. 13th ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst. PODS*, Minneapolis, MN, USA, 1994, pp. 77–85.
[95] J. Kivinen and H. Mannila, "Approximate dependency inference from relations," in *Proc. Int. Conf. Database Theory*, Berlin, Germany, Oct. 1992, pp. 86–98.
[96] J. Liu, J. Li, C. Liu, and Y. Chen, "Discover dependencies from data—A review," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 251–264, Feb. 2012.
[97] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for data cleaning," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 746–755.
[98] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Trans. Database Syst.*, vol. 33, no. 2, pp. 1–48, Jun. 2008.
[99] F. D. Marchi, S. Lopes, and J.-M. Petit, "Efficient algorithms for mining inclusion dependencies," in *Proc. Int. Conf. Extending Database Technol.*, Prague, Czech Republic, Mar. 2002, pp. 464–476.
[100] F. D. Marchi, S. Lopes, and J.-M. Petit, "Unary and n-ary inclusion dependency discovery in relational databases," *J. Intell. Inf. Syst.*, vol. 32, no. 1, pp. 53–73, Feb. 2009.
[101] T. Papenbrock, S. Kruse, J.-A. Quiané-Ruiz, and F. Naumann, "Divide & conquer-based inclusion dependency discovery," *Proc. VLDB Endowment*, vol. 8, no. 7, pp. 774–785, Feb. 2015.
[102] X. Zhu, S. Song, X. Lian, J. Wang, and L. Zou, "Matching heterogeneous event data," in *Proc. Int. Conf. Manage. Data, SIGMOD*, Snowbird, UT, USA, Jun. 2014, pp. 1211–1222.
[103] S. Song, Y. Cao, and J. Wang, "Cleaning timestamps with temporal constraints," *Proc. VLDB Endowment*, vol. 9, no. 10, pp. 708–719, Jun. 2016.
[104] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava, "Sequential dependencies," *Proc. PVLDB*, vol. 2, no. 1, pp. 574–585, 2009.
[105] J. Wang, S. Song, X. Zhu, and X. Lin, "Efficient recovery of missing events," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 841–852, Aug. 2013.
[106] J. Wang, S. Song, X. Zhu, X. Lin, and J. Sun, "Efficient recovery of missing events," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 2943–2957, Jul. 2016.
[107] J. Wang, S. Song, X. Lin, X. Zhu, and J. Pei, "Cleaning structured event logs: A graph repair approach," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, Apr. 2015, pp. 30–41.
[108] W. Fan, Z. Fan, C. Tian, and X. L. Dong, "Keys for graphs," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1590–1601, Aug. 2015.
[109] W. Fan, C. Hu, X. Liu, and P. Lu, "Discovering graph functional dependencies," in *Proc. Int. Conf. Manage. Data SIGMOD*, Houston, TX, USA, 2018, pp. 427–439.
[110] S. Song and L. Chen, "Discovering matching dependencies," in *Proc. 18th ACM Conf. Inf. Knowl. Manage. CIKM*, Hong Kong, 2009, pp. 1421–1424.
[111] S. Song and L. Chen, "Efficient discovery of similarity constraints for matching dependencies," *Data Knowl. Eng.*, vol. 87, pp. 146–166, Sep. 2013.
[112] W. Fan, X. Jia, J. Li, and S. Ma, "Reasoning about record matching rules," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 407–418, Aug. 2009.
[113] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma, "Dynamic constraints for record matching," *VLDB J.*, vol. 20, no. 4, pp. 495–520, Aug. 2011.
[114] S. Song, L. Chen, and H. Cheng, "On concise set of relative candidate keys," *Proc. VLDB Endowment*, vol. 7, no. 12, pp. 1179–1190, Aug. 2014.
[115] Y. Wang, S. Song, L. Chen, J. X. Yu, and H. Cheng, "Discovering conditional matching rules," *ACM Trans. Knowl. Discovery Data*, vol. 11, no. 4, pp. 1–38, Aug. 2017.
[116] S. Kwashie, J. Liu, J. Li, and F. Ye, "Efficient discovery of differential dependencies through association rules mining," in *Proc. Australas. Database Conf.* in Lecture Notes in Computer Science, vol. 9093, M. A. Sharaf, M. A. Cheema, and J. Qi, Eds. Melbourne, VIC, Australia: Springer, Jun. 2015, pp. 3–15.



ZHICHENG LIU received the B.E. degree from the School of Software, Harbin Institute of Technology, China, in 2018. He is currently pursuing the M.E. degree with the School of Software, Tsinghua University, China. His research interest includes data quality analytics.

AOQIAN ZHANG received the B.E. degree in computer software and the Ph.D. degree in software engineering from the School of Software, Tsinghua University, in 2013 and 2018, respectively. He is currently a Postdoctoral Fellow with the Data System Group, Cheriton School of Computer Science, University of Waterloo.

• • •