EPiC Series in Computing

Volume 98, 2024, Pages 46–55

Proceedings of 39th International Conference on Computers and Their Applications

Error Assessment for Multi-Join AQP using Bootstrap Sampling

Sabin Maharjan[1], Lucy Kerns[2], Xiangjia Min[3], and Feng Yu[1]*

[1] Computer Science and Information Systems, Youngstown State University, Youngstown, Ohio, USA
smaharjan@student.ysu.edu,fyu@ysu.edu

[2] Statistics and Mathematics, Youngstown State University, Youngstown, Ohio, USA
xlu@ysu.edu

[3] Chemical and Biological Sciences, Youngstown State University, Youngstown, Ohio, USA
xmin@ysu.edu

Abstract

Approximate query processing (AQP) is a computing efficient scheme to provide fast and accurate estimations for big data queries. However, assessing the error of an AQP estimation remains an open challenge for high-dimensional multi-relation data. Existing research often focuses on the online AQP methods which heavily rely on expensive auxiliary data structures. The contribution of this research is three-fold. First, we develop a new framework employing a non-parametric statistic method, namely bootstrap sampling, towards error assessment for multi-join AQP query estimation. Second, we extend the current AQP schemes from providing point estimations to range estimations by offering the confidence intervals of a query estimation. Third, a prototype system is implemented to benchmark the proposed framework. The experimental results demonstrate the prototype system generates accurate confidence intervals for various join query estimations.

1 Introduction

A demanding challenge in the big data era is to answer complex queries given a time limitation. Much research has been developed to promptly provide the exact answer for data queries [12, 13]. However, getting the exactly precise answer for every data query is not always necessary in all circumstances. For example, during the exploratory data analysis (EDA), a user often prefers a quick query approximation without blinking of eyes rather than the exact answer coming after a long waiting. Many complex big data queries require prolonged waiting time and waste significant computing energy.

Approximate query processing (or AQP) is an alternative scheme that aims to quickly provide estimated query answers with satisfying accuracy and usually is given a short time restriction [3, 10, 11]. Without the need to execute the query on the original dataset, which can be time-consuming, AQP only collects a statistical summary of the data, named synopsis,

*Corresponding Author

and runs the query on the synopsis to obtain a synopsis query result. A query estimation is generated from the synopsis query result using AQP estimators. AQP can provide a query estimation much faster than traditional query processing schemes [7].

Based on the schemes of statistics collection, AQP can be categorized into two groups, namely the online AQP and offline AQP. The *online AQP* [5, 8, 9] collects statistical summaries only after the target query for approximation is submitted. To achieve a fast responding speed, the online AQP heavily relies on auxiliary data structures such as indices and hash tables to scan data quickly. These data structures are usually costly to maintain and time-consuming to construct. Another drawback is the statistics collected by online AQP are not reusable for a different target query. These drawbacks lead to significant waste of computing energy and storage space.

The *offline AQP*[2, 14], on the other hand, collects a priori statistics before a query is submitted for estimation. It typically creates a holistic statistical synopsis based on the schema of the database. A particular advantage of the offline AQP is it doesn't rely on expensive auxiliary data structures. Another advantage is the collected statistics by the offline AQP are reusable for future queries. These significantly reduce the overall system cost compared with the online AQP.

One current challenge is to assess the accuracy of AQP query estimations, namely error estimation [4]. The difficulty arises from the phenomena that, when the query conditions change, the underlying distributions of the result data are also changed and difficult to predict. For online AQP schemes, the statistics are collected on-the-fly after the query result is obtained; therefore, the ground truth query result distribution is obtained, which alleviates the problem complexity for the online AQP. However, for offline AQP schemes, the ground truth query result distribution is not collected on-the-fly and hence unknown. This produces a major obstacle of error estimation for online AQP schemes.

Bootstrap sampling [6] is a unique statistical technique that can assess the errors of a sample-based estimator. One advantage of bootstrap sampling is that it doesn't require prior knowledge of the ground truth population distribution, but can *pull itself up from its bootstrap*. Therefore, it is often considered as a *non-parametric statistical method*. Bootstrap sampling performs a special sampling method, namely resampling, which can generate a large number of *bootstrap replications* in order to estimate the standard deviation (or standard error) of an AQP query estimator.

In this work, we study the problem of using bootstrap sampling to assess estimation errors for offline join-AQP systems answering multi-join query results. Our contributions include the follows. First, we propose a framework that integrates bootstrap sampling with a sample-based offline AQP scheme, namely CS2 [14], which can provide error assessments for the query estimator. Second, the proposed framework will generate a confidence interval for the query estimation and extend an AQP system from providing point-estimations to range estimations for query results. Third, a prototype system implemented to simulate a real-world database system. This system is equipped with a bootstrap sampler engine enabling error assessment for AQP schemes. We test the performance of the prototype system on multiple datasets with various combinations of hyperparameter settings. The experimental results demonstrate the confidence intervals produced reach high accuracy for various test queries.

The rest of the work is organized as follows. Section 2 introduces the background of AQP and bootstrap sampling. Section 3 formulates the problem of join AQP error estimation. Section 4 describes error assessment for the sampled-based join AQP scheme. We describe the implementation of the prototype system enabling bootstrap sampling working with join AQP in Section 5. Experimental results are discussed in Section 6. Section 7 includes the conclusion

and future work.

2 Background

Approximate query processing (or AQP) is a technology to provide rapid estimations for complex queries using statistical methods. It aims to provide accurate query estimations within a short time frame.

2.1 AQP for multi-relation join queries

The multi-relation join (or multi-join) approximate query processing focuses on estimating queries including multiple joins and selections on several joinable tables or correlated datasets. Given a multi-join query Q on a table R, to get the ground truth query result Y_{GT}, the traditional query processing techniques require executing the query Q on the original dataset $\mathcal{R} = \{R_i\}_{i=1}^{n}$ which can take a long time without relying on high-performance computing hardware or costly data structures such as indices and hash tables.

Instead of running the query Q directly on the original dataset $\mathcal{R}$, the multi-join AQP usually executes the query Q on a synopsis $\mathcal{S}$ generated from $\mathcal{R}$ in order to produce an estimated answer of the query result. A synopsis is typically a statistical summary of the original dataset $\mathcal{R}$. In a simple example, for an equi-join query counting the size of join between R_1 and R_2, $Q = |R_1 \bowtie R_2|$. If there is a simple random sample without replacement (SRSOR) on R_1 denoted by S_1. Y_s is the sample query result after running Q on S_1. The ground truth Y_{GT} can be estimated by $\hat{Y} = \frac{Y_s}{f}$ where $f = \frac{|S|}{|R|}$ is the sampling ratio.

2.2 Bootstrap Sampling

Bootstrap sampling performs a unique process named *resampling* or *sampling with replacement*. Each iteration of this procedure generates a new distribution, namely a *bootstrap sample*, which is a simple random sample with replacement (SRSWR) from the original sample dataset. Applying desired statistics or functions on a resampled distribution, a scalar is computed named the *bootstrap replication*. Bootstrap resampling usually generates a large number of bootstrap replications and use them to estimate useful statistical features, such as the standard deviation of the original dataset even when the population (or ground truth) distribution is unknown.

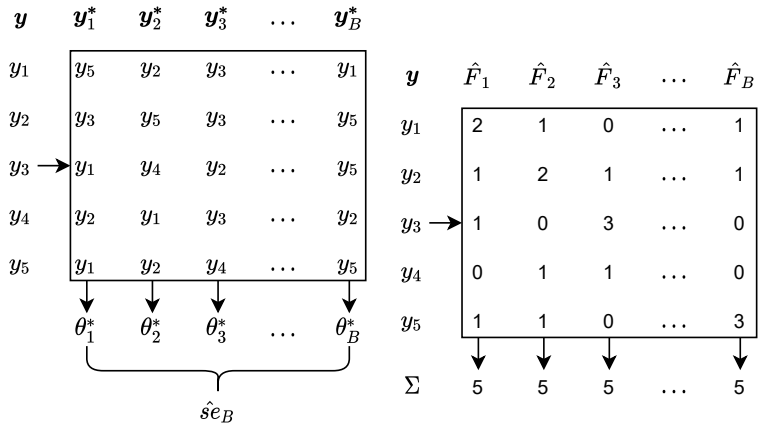
Figure 1 depicts a sample example of how bootstrap sampling is performed. When given a sample dataset $\vec{y} = (y_i), i = 1, ..., n$ from an unknown distribution F, a *bootstrap sample* $\vec{y}^* = (y_i^*), i = 1, ..., n$ is a collection obtained by n times of SRSWR the original sample $\{y_i\}_{i=1}^{n}$. For instance, if $n = 5$, we might obtain different bootstrap samples, such as $\vec{y}_1^* = (y_5, y_3, y_1, y_2, y_1)$, $\vec{y}_2^* = (y_2, y_5, y_4, y_1, y_2)$, $\vec{y}_3^* = (y_3, y_3, y_2, y_3, y_4)$, etc. These resamples are shown in Figure 1a.

After summarizing the frequency of each sampled element, we obtain the distribution of a bootstrap sample, $\hat{F} = (\hat{f}_1, \hat{f}_2, \dots)$, where $\hat{f}_k = \#\{y_i^* = y_k\}/n$.

2.2.1 An example application of bootstrap sampling

An example application using bootstrap sampling is to estimate the standard deviation (or standard error) s of a sample estimator from an unknown distribution.

Suppose we are interested in a statistical parameter $\theta = t(F)$ where t is a statistical or analytic function. Since usually we don't have all the information of F, but can only calculate an estimation of θ from a given sample $\vec{y}$ denoted by $\hat{\theta} = s(\vec{y})$. For each bootstrap sample $\vec{y}^*$



(a) Bootstrap Samples (b) Bootstrap Distributions

Figure 1: Example: Bootstrap Sampling

we can generate a bootstrap replication of $\hat{\theta}$, denoted by $\hat{\theta}^* = s(\vec{y}^*)$. For instance, when $\hat{\theta}$ is the sample mean $\overline{\vec{y}}$, a bootstrap replication $\hat{\theta}^*$ is be the sample mean on a bootstrap sample $\overline{\vec{y}^*}$.

After generating a total number, B, of bootstrap samples, we can obtain the standard deviation of all $\hat{\theta}^*$, i.e. $\widehat{se}_B(\hat{\theta}^*)$, called the *bootstrap estimation of standard error*. Generally speaking, the large the number B, the better $\widehat{se}_B(\hat{\theta}^*)$ estimates the ground truth standard deviation of $\hat{\theta}$, $se_F(\hat{\theta})$.

Often, $se_{\hat{F}}(\hat{\theta}^*)$ is called a *plug-in* estimate of $se_F(\hat{\theta})$ that uses the empirical distribution $\hat{F}$ in replacement of the population distribution F.

$\widehat{se}_B(\hat{\theta}^*)$ can be calculated as

$$\widehat{se}_B(\hat{\theta}^*) = \left[\frac{1}{B-1} \sum_{i=1}^{B} \left(\hat{\theta}^*(i) - \bar{\theta}^* \right)^2 \right]^{\frac{1}{2}} \tag{1}$$

where $\bar{\theta}^* = \sum_{i=1}^{B} \hat{\theta}^*(i)/B$.

3 Problem Statement

We consider the following formulation for a multi-join query Q in this research.

```
SELECT Fn(attribute collection) FROM table collection WHERE conditions;
```

where Fn is a common and "smooth" aggregate function such as COUNT, AVG, and SUM as it's well known that non-smooth aggregate functions, for example median and count distinct, do not suite for sample-based AQP frameworks; table collection is the collection of all tables involved in the query (denoted by $\mathcal{R}^Q = \{R_i^Q\}_{i=1}^m$); attribute collection includes attributes $\{A_{ij}^Q\}$ displayed in the query result; the conditions includes the selection conditions and join conditions.

4 Methodology

4.1 Multi-Join Query Result Estimation

Suppose a synopsis $\mathcal{S}$ is constructed on the original dataset $R = \{R_i\}_{i=1}^{M}$, and the top relation in the join graph of $\mathcal{R}^Q$ is R_1^Q and its corresponding sample table is S_1^Q. When a query Q is executed on the synopsis $\mathcal{S}$, each sample tuple $u_i \in S_1^Q$, $i = 1, ..., n$, produces a tuple query result y_i according to the aggregate function Fn and the join result. Let $S_Q = \{y_i\}_{i=1}^{n}$ denote the collection of tuple query results. For instance, if Fn is COUNT and the primary key is included in the attribute collection then y_i is either 1 if the tuple u_i satisfies the query conditions or otherwise is 0.

The query result Y_s of Q on $\mathcal{S}$ is calculated as $Y_s = \sum_{i=1}^{n} y_i$ where $n = |S_1^Q|$. Suppose $N = |R_1^Q|$ and the sample fraction $f = \frac{n}{N}$, then the estimation of the ground truth of query result on $\mathcal{R}$, Y_{gt} is $\hat{Y} = \frac{Y_s}{f}$.

The accuracy of $\hat{Y}$ depends on both the sampling method and the statistical nature of the tuple query result set $Y = \{Y_i\}_{i=1}^{N}$ produced by each tuple $U_i \in R^{Qt}$ when executing query Q on $\mathcal{R}$. The accuracy of $\hat{Y}$ tends to be higher when the sampling process is uniform and the skewness of Y is low; otherwise, the it will be lower when the sampling process is not uniform or Y has a high skewness or even includes correlation.

4.2 Bootstrap Sampling

First, the tuple query results $S_Q = \{y_i\}_{i=1}^{n}$ are obtained by executing Q on the synopsis $\mathcal{S}$. Then $\{\vec{y}_j\}_{j=1}^{B}$ is generated via bootstrap re-sampling on S_Q for a total of B iterations. Each $\vec{y}_j = \{y_{j,i}\}_{i=1}^{n}$ is a bootstrap re-sample of S_Q where each tuple query result $y_{j,i}$ is randomly sampled with replacement from S_Q.

The bootstrap replication $\hat{Y}_j$ is the estimation of Y_{GT} using the query result. Let $Y_{\vec{y}_j}$ be the query result aggregated from $\vec{y}_j$. It can be calculated as $\hat{Y}_j = \frac{Y_{\vec{y}_j}}{f}$. For example, if the aggregate function is COUNT, then the estimation is $\hat{Y}_j = \frac{1}{f}\sum_{i=1}^{n} y_{j,i}$.

The collection of all B bootstrap replication is denoted by $\hat{Y}_B = \{\hat{Y}_j\}_{j=1}^{B}$. The standard deviation of $\hat{Y}_B$ is

$$\hat{se}_B = \left[\frac{1}{B-1}\sum_{j=1}^{B}(\hat{Y}_j - \bar{\hat{Y}}_B)^2\right]^{\frac{1}{2}} \tag{2}$$

where $\bar{\hat{Y}}_B$ is the sample mean of all bootstrap replications $\hat{Y}_B$. By the theoretical framework of bootstrap sampling, we claim the Eq 2 is the *bootstrap estimation of the standard error* of the $\hat{Y}$ when estimating Y_{GT} for query Q. The accuracy of $\hat{se}_B$ tends to be higher when B increases. Usually we consider B of 1000 to 2000 as sufficient.

4.3 Bootstrap Confidence Interval

There are multiple methods to calculate the confidence interval (or CI) using bootstrap sampling, such as the standard method and percentile method. These methods are usually efficient and fairly accurate. There are also improved methods to increase the accuracy and adjust the bias such as BCa and ABC methods; however, they are often slower in performance

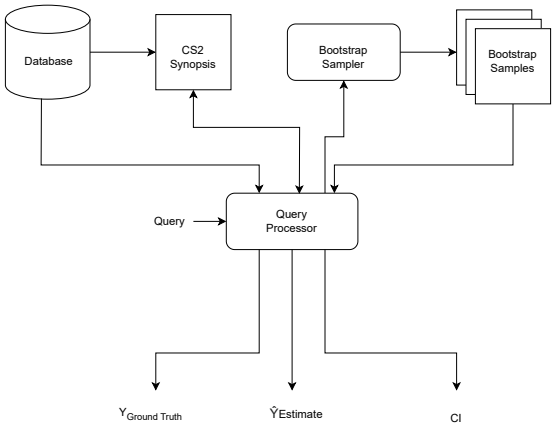


Figure 2: Prototype AQP System Architecture

for larger datasets [6]. In this work, we employ the commonly adopted standard method because it's a generally applied method and our aim is to investigate the overall system performance on large datasets.

Suppose the significant level is denoted by α which is a probabilistic value. Common choices of α include 5% for 90% level of significance and 2.5% for 95% level of confidence, respectively. The bootstrap CI calculated by the standard method is as follows.

$$\left(\hat{Y} - z^{(1-\alpha)} \cdot \hat{se}_B, \hat{Y} + z^{(1-\alpha)} \cdot \hat{se}_B\right) \tag{3}$$

where $\hat{Y}$ is the query estimation and $z^{(1-\alpha)}$ is the $100(1-\alpha)$th percentile of a standard normal distribution. For example, for 90% level of significance, $z^{(0.95)} = 1.645$, and for 95% level of significance, $z^{(.975)} = 1.960$.

5 Prototype AQP System Implementation

Figure 2 depicts the architecture of the prototype AQP system with bootstrap sampling module. The major components include a simple query parser, a query processor, CS2 synopsis generator, and the bootstrap sampler. The query parser processes SQL queries stored in a plain textfile and generates query structures. The query processor executes an SQL query based on the query structure and computes the ground truth query result Y_{GT}. The current query processor is cable to process simple SQL opperators such as selection, join, and filtering conditions (in the WHERE clause).

The CS2 synopsis generator can perform correlated sampling on the original database. The created CS2 synopsis will be fed into the query processor which will produce sample query results and hence generate the AQP estimation ($\hat{Y}$) for a SQL query. The sample query results will be given to the Bootstrap sampler to generate bootstrap resamples which will be used to estimate the standard deviation of the AQP query estimation. The standard deviation will later used to produce the confidence interval (CI) of the AQP query estimation. The CI will be compared with the ground truth query result in order to evluate the accuracy of Bootstrap sampling for AQP error estimation.

The prototype system is developed in Rust to mimic a realistic system when processing queries on large datasets.

6 Experiment

6.1 Experiment Setup

The experiment was performed on a server equipped with an Intel(R) Core(TM) i7-10710U CPU which operates at a base frequency of 1.10GHz. The system is equipped with 21GB of RAM and runs the operating system of CentOS 7 with the Linux kernel version 3.10. The experiment code is developed using the Python 3 programming language which runs the prototype system implemented in Rust.

The datasets used for experiments are generated using TPC-H benchmark [1] in different data sizes including 100MB, 1GB, and 10GB stored on a centralized hard drive on the server. The test queries are created from the TPC-H queries with randomly generated values in the filtering conditions. The queries are grouped by the number of joins which ranges from one to four joins, where each group includes 10 test queries. To obtain reliable running results, each test query is executed for 10 times and retain the averaged running result.

When creating the CS2 synopses, the sampling fractions used include 0.1%, 0.5%, and 1.0% mimicking the scenarios with small, medium, and large sampling sizes. In the bootstrap sampling process, the total number of bootstrap resamples used include B=200 and B=2000 to test its impact on the accuracy of error estimation.

In general, it is observed that the average hit percentages are above 90% across all figures mentioned. The hit percentages are observed fairly accurate even given small sample fractions and few bootstrap resample iterations.

6.2 Accuracy Tests

The bootstrap sampling model in the experimental system will produce the standard deviation and hence a confidence interval (CI) of the AQP query estimation. The optimal situation is the ground truth query result obtained using the original database is included between the upper and lower bound of the CI, which we call it the *hit* scenario; otherwise, it's a *miss* scenario. We introduce an easy understanding measure called *hit percentage* (or *hit ratio*) to evaluate the accuracy of the bootstrap sampling model described as follows.

$$\text{hit percentage} = \frac{\text{times(CI hits)}}{\text{times(total experiments)}} \times 100\% \tag{4}$$

We compute the hit percentage for test queries in each group of join numbers. We analyze the impact of hyperparameters, such as sampling fractions and bootstrap resample iterations, on the hit percentage across various sizes of datasets and query groups.

6.3 Factor of Sampling Fractions

Figures 3, 4, and 5 depict the average hit percentages given various sample fractions (or sample size, f) in each group of join query and bootstrap resample iteration (B). We generated CS2 synopses with sampling factions set to 0.1%, 0.5%, and 1.0% on each of the dataset. We also grouped the hit percentage results according to B is set to 200 and 2000.

We observe that the CS2 synopsis volumes progressively increase given larger sample fractions. A larger synopsis includes more samples from the original raw data and will improve the accuracy of AQP estimations. Therefore, when set the B value fixed, the hit percentage generally increases when the sample fraction increases.

Another perspective is the changing of join numbers. When join numbers increase, the AQP estimation tends to be lower as the query become more complex and the sample join results may fluctuate. Therefore, the hit percentage tends to be lower when the join number increases.

6.4 Factor of Bootstrap Resample Iterations

Figures 6, 7, and 8 depict how the average hit percentages change given different bootstrap resample iterations in each group of join query and sample fraction.

Provided more iterations of bootstrap resampling, more bootstrap replications will be produced to better estimate the standard deviation of the AQP estimation. Therefore, setting the sf value fixed, the hit percentage generally increases when more bootstrap resamples are computed. Same as previously mentioned, the hit percentage tends to be lower when the join number increases.

7 Conclusion

In this work, we introduced a non-parametric framework of error assessment for the offline join AQP scheme using bootstrap sampling. The contributions are twofold. First, we developed the methodology to perform the bootstrap sampling of join query results on a sample-based synopsis. Then we provided the theoretical framework to calculate the confidence interval of the AQP estimation. This improves the traditional AQP point estimator to a range estimate with a significant improvement in the usability of the AQP application. The second contribution is we implemented a prototype system that integrates the bootstrap sampler with a join-AQP framework. Extensive experiments have been performed on the prototype system using multiple datasets and test queries. The experimental results showed that the prototype system produced satisfying accuracy in error assessment for join AQP estimations. In the future, we will generalize the framework for more challenging AQP scenarios, such as when correlation exists in the statistical synopsis.

References

[1] Tpc-h benchmark. https://www.tpc.org/tpch/.

[2] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. pages 275–286. ACM, 1999.

[3] Sameer Agarwal, Henry Milner, Ariel Kleiner, Ameet Talwalkar, Michael Jordan, Samuel Madden, Barzan Mozafari, and Ion Stoica. Knowing when you're wrong: Building fast and reliable approximate query processing systems. pages 481–492, 2014.

[4] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. pages 511–519, 2017.

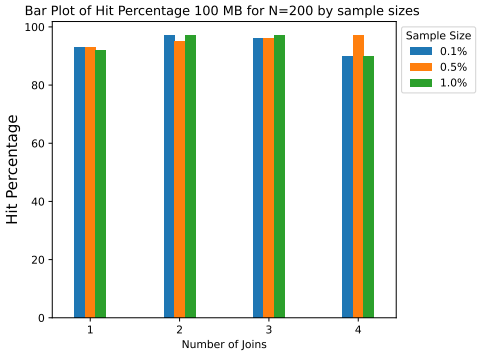
[5] Yu Chen and Ke Yi. Two-level sampling for join size estimation. pages 759–774. ACM, 2017.

[6] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[7] Christopher Jermaine, Subramanian Arumugam, Abhijit Pol, and Alin Dobra. Scalable approximate query processing with the dbo engine. *ACM Trans. Database Syst.*, 33:1–54, 2008.

[8] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. Cardinality estimation done right : Index-based join sampling. 2017.

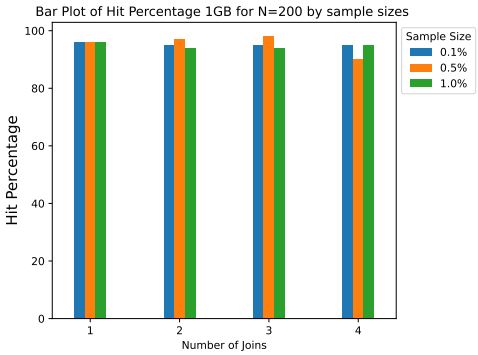
[9] Feifei Li, Bin Wu, Ke Yi, Zhuoyue Zhao, Lihong Li, Shawna Miles, Zephan Melville, Amalthiya Prasad, and Linda L Breeden. Wander join: Online aggregation via random walks. *Proc. SIGMOD'16*, pages 615–629, 2016.



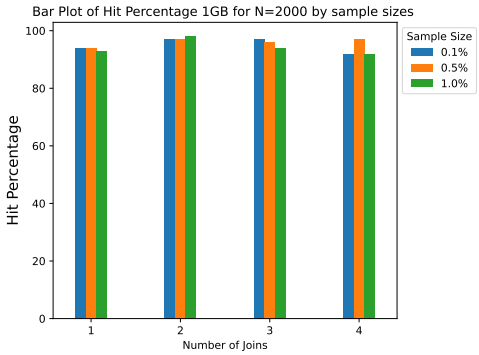
(a) 100MB, B=200

(b) 100MB, B=2000

Figure 3: Hit percentage for 100 MB data set with different bootstrap samples

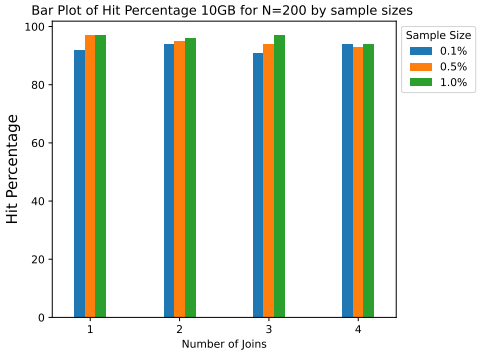


(a) 1GB, B=200

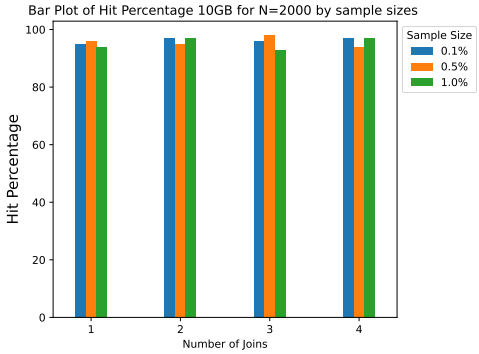


(b) 1GB, B=2000

Figure 4: Hit percentage for 1GB data set with different bootstrap samples

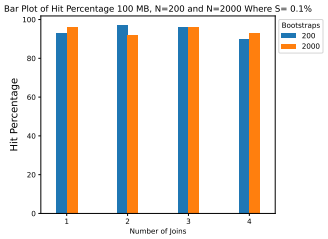


(a) 10GB, B=200

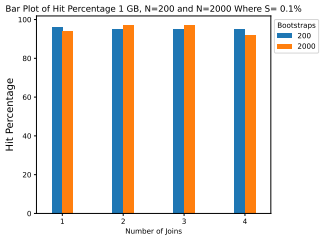


(b) 10GB, B=2000

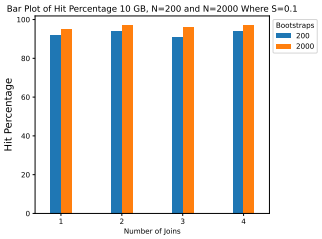
Figure 5: Hit percentage for 10GB data set with different bootstrap samples



(a) 100 MB S=0.1%

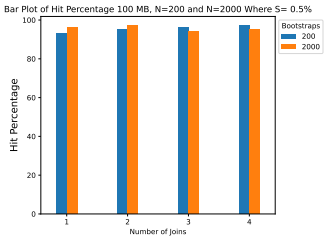


(b) 1 GB S=0.1%

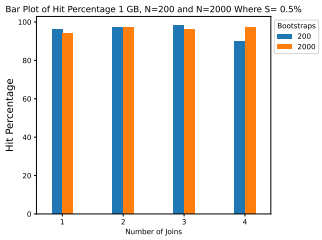


(c) 10GB S=0.1%

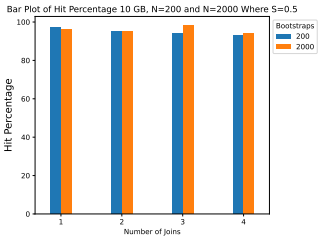
Figure 6: Hit percentage for 0.1% sample size and different data sizes



(a) 100 MB S=0.5%

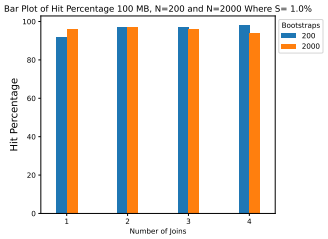


(b) 1 GB S=0.5%

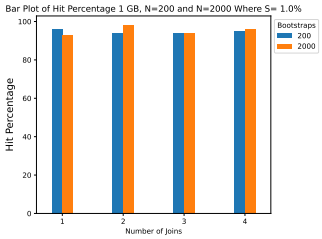


(c) 10GB S=0.5%

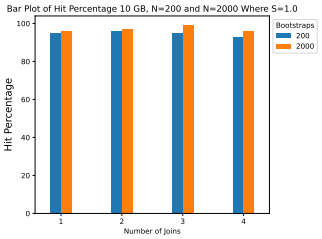
Figure 7: Hit percentage for 0.5% sample size and different data sizes



(a) 100 MB S=1.0%



(b) 1 GB S=1.0%



(c) 10GB S=1.0%

Figure 8: Hit percentage for 1% sample size and different data sizes

[10] Kaiyu Li and Guoliang Li. Approximate query processing: what is new and where to go? *Data Science and Engineering*, 3:379–397, 2018.

[11] Qing Liu. Approximate query processing. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 113–119. Springer US, 2009.

[12] D Le Quoc, I Ekin Akkus, Pramod Bhatotia, Spyros Blanas, Ruichuan Chen, Christof Fetzer, Thorsten Strufe, Do Le Quoc, Istemi Ekin Akkus, Pramod Bhatotia, Spyros Blanas, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. Approximate distributed joins in apache spark. *ArXiv e-prints*, abs/1805.0, 5 2018.

[13] Marc Sch, Johannes Schildgen, and Stefan Deßloch. Sampling with incremental mapreduce. 2015.

[14] Feng Yu, Wen-Chi Hou, Cheng Luo, Dunren Che, and Mengxia Zhu. CS2: A new database synopsis for query estimation. pages 469–480. ACM, 2013.