International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 11 (2018) pp. 9420-9427
© Research India Publications. http://www.ripublication.com

Efficient Execution of Aggregate Queries on Big Data

Malatesh. S. Havanur*[1], Y.S. Kumarswamy[2]

[1]*Research Scholar, Department of Computer Science& Engineering., Rayalaseema University, Kurnool, 518007, Andra Pradesh, India.*

[2]*Professor, Dean R&D, Department of Computer Science& Engineering., NCET, Bengaluru,Karnataka India.*

Abstract

The term Big Data, vaguely refers to huge volumes of data generated at high velocity. Currently, many enterprises are generating Big Data, which may contain high impact knowledge. Knowledge mining of Big Data has become essential for many enterprises in-order to predict the viability of future business decisions. One of the important components of knowledge mining is aggregate query. Execution of aggregate queries on Big Data can lead to explosion in response time. Hence, sampling techniques which provide approximate answers in limited time to aggregate queries are much preferred. However, implementing efficient sampling techniques is hindered due to unstructured data and lack of indexes. The existing treatment in the literature for this sampling issue in Big Data is limited, and lacks the required effectiveness. In this work, a new sampling scheme for approximate aggregate query execution on Big Data is proposed, which utilizes the Map Reduce model, and overcomes the major sampling issues associated with Big Data. Four aggregate operations--sum, average, variance and standard deviation are addressed. The query result estimators are built by using Central Limit Theorem. The empirical results exhibit appreciable accuracy of the proposed approximate query execution techniques.

Keywords: Sampling; Big Data; Aggregate Query; Central Limit Theorem

INTRODUCTION

Currently, the enterprise data is burgeoning due to data production by numerous gadgets such as--camera, mobile devices, remote sensing devices, social networking systems, wireless sensor networks etc. This gargantuan data is also produced at high velocity. This high velocity, usually unstructured and high volume data is vaguely referred as Big Data.

Many enterprises require effective prediction mechanism to execute future business decisions with minimal risk. Knowledge mining of Big Data is one of the effective prediction tools available for enterprises. The task of knowledge mining involves execution of multiple queries, and aggregate queries are one of the most important aspects of knowledge mining. However, aggregate queries consume unattractive execution latency, and execution of multiple aggregate queries can compound the unattractiveness. Hence, sampling techniques seem an attractive option to reduce the execution latency, and provide approximate answers with good accuracy.

Overview on Sampling Techniques

Let, n represent the size of a given population, θ represent a statistic which needs to be calculated by using all the objects of the population, and $\hat{\theta}$ indicate the estimated value of θ, which is calculated as represented in Equation 1. Here, $est()$ indicates the estimator function, $(o_1, o_2, \ldots\ldots o_m)$ indicate randomly selected objects with replacement from the population, and $m < n$. Each object $o_i \in$ population being selected is equally likely as represented in Equation 2. In this work the concept of consistent estimator represented in Equation 3 will be utilized.

$$\hat{\theta} = est(\theta|(o_1, o_2, \ldots\ldots o_m), \mathrm{n}) \quad (1)$$

$$P(o_i) = \frac{1}{n} \quad (2)$$

$$\lim_{m \to n} est(\theta|(o_1, o_2, \ldots\ldots o_m), \mathrm{n}) = \theta \quad (3)$$

Motivation

It is clear from Section 1.1 that, the sampling techniques usually require random access of objects in the population, and in some cases the size of population is also required to be known. These two constraints are well satisfied in the structured data processing systems through the aid of indexes and metadata statistics. However, in the unstructured environment--usually found in Big Data--the required facilities to satisfy these two constraints are usually not present, and difficult to implement. Also, concurrent structuring of data is difficult due to high velocity of data production.

Until now, in the literature, these two constraints have not been effectively addressed. The goal of this work is to present an approximate query execution technique for aggregate queries, which utilizes sampling scheme by resolving the said sampling constraints on Big Data.

Contributions

The following contributions are made in this work:

1. Four aggregate operations--sum, average, variance and standard deviation are considered for the proposed approximate query processing technique, which is implemented inside the Map Reduce framework. A randomization scheme is presented to facilitate random access of unstructured data, and generate the required metadata statistics. The sampling formulations are designed based on Central Limit Theorem (CLT).
2. The proposed approximate query processing technique is evaluated over TPC-DS data set. Empirical results exhibit the excellent performance of the proposed technique w.r.t. accuracy and execution efficiency.

Paper Organization

This paper is organized as follows: Section 2 describes the related work. Section 3 presents the proposed approximate query processing technique for aggregate queries. Section 4 presents the empirical results and corresponding discussions. Finally, the work is concluded in Section 5.

RELATED WORK

Sampling techniques for approximate query processing in RDBMS has been extensively studied in the literature [1-19].

The adaptive sampling technique [9] utilizes a stopping condition, which does not depend upon number of samples, and depends upon the size of samples. If the sample sizes are large, then, the stopping condition can be fulfilled with very few samples. However, indexes are required for its efficient implementation.

Sequential sampling [15] utilizes block data access, and existing sampled information. This technique requires block data access for its effective implementation, and this requirement might not be realized in all the platforms.

The Guaranteed Error Estimator [11], proposes a sampling scheme, and bounded error formulation for estimating the number of distinct values present in the attribute.

It was argued in [1] that, instead of providing specific error result approximation for aggregate queries, range of approximate solutions with varying error would be much more attractive. This presented technique [1] is denoted as Online Aggregation. During the execution of an aggregate query, continuous approximate solutions with varying error are provided, and the user can stop the current query execution when satisfiable approximate solution is discovered.

The random sampling over joins [17] technique optimizes the execution efficiency of collecting random samples for a join query by pushing the sampling operator below the join node in the query plan tree.

Multiple contributions [20-29] have been made to extend the Map Reduce framework to include sampling techniques. For example, Oracle RAC, Teradata and Netezza provide facility to execute queries on sample data. However, these platforms provide no error guaranteed solution.

It was observed in [20] that, production of random samples in Map Reduce framework can lead to poor efficiency. Hence, a new framework to address this problem was proposed, which divides the input split into numerous sub-splits. Each sub-split is randomly chosen to produce random samples. However, this work did not address the persistent problems as mentioned before.

It is clear from the described related work that, the existing efforts have concentrated on extending the Map Reduce framework for sampling schemes. However, the problems addressed in this work have not been effectively resolved.

APPROXIMATION TECHNIQUE FOR AGGREGATE QUERIES

Problem Statement

The proposed technique requires block level access of data, which can be efficiently achieved by using existing features of Map Reduce framework. Let, $m(m > 0)$ blocks of qualifying tuples for Q exist in a particular relation on which, the aggregate query$Q$ is executed, wherein, m is very large. The data is assumed to be unstructured, and the potential to be structured exists. Let, A be the specific attribute on which Q is executed. The number of tuples $\in block\ B_i (1 \leq i \leq m)$ is indicated by n_i . Also, Q represents a single aggregate operation query which can contain any of the four aggregate operations namely--sum, variance, average and standard deviation.

Let, $R(Q)$ indicate the result of Q. The goal is to design an estimator for $R(Q)$, which satisfies the condition represented in Equation 4. Here, $p,\ est(Q)$and $\in$ indicate the estimator confidence, the estimator for $R(Q)$ and estimation error respectively.

$$P(|R(Q) - est(Q)| \leq \in) = p \qquad (4)$$

Randomization Scheme

The randomization scheme is built over the Map Reduce framework. Block level access of tuples is performed to calculate $est(Q)$. Each block is assigned to a Map task. In-order to obtain a single random sample/tuple, one of the available blocks will be randomly selected, wherein, each block is equally likely to be selected, and the corresponding Map task performs structuring of block tuples. From the selected block, one of the tuple inside the block is randomly selected, wherein, each tuple is equally likely to be selected.

The proposed randomization scheme, clearly, overcomes the tuple dependency issue for the block tuples. Since, each block

contains very few qualifying tuples, the randomization scheme is also not resource intensive.

Sampling Scheme

The number of tuples which satisfy *Q* is indicated by *N,* which is assumed very large, and it is calculated as represented in Equation 5. In-order to calculate *N*, all the *m* blocks have to be selected, so, *N* will be estimated, and the corresponding details will be subsequently outlined.

$$N = \sum_{i=1}^{m} n_i \quad (5)$$

Let, the values of random samples corresponding to *A* be indicated as $[v_1, v_2, \dots v_n]$, and since, these values are independently and equally likely produced, they can be considered as IID observations.

The CLT is represented in Equation 6. Here, *SN* indicates the standard normal distribution, μ indicates the mean of the random samples, since, the sample mean represented in Equation 7 is a consistent estimator as represented in Equation 8, the sample mean indicated by $\bar{v}_n$ can be used as an estimator for μ, σ^2 indicates the variance of random samples, and since the sample variance represented in Equation 9 is a consistent estimator as represented in Equation 10, the sample variance indicated by $\bar{S}_n$ can be used as an estimator for σ^2.

$$P(|\bar{v}_n - \mu| \leq \epsilon) \approx 2SN\left(\frac{\epsilon\sqrt{n}}{\sigma}\right) - 1 \quad (6)$$

$$\bar{v}_n = \frac{\sum_{i=1}^{n} v_i}{n} \quad (7)$$

$$\lim_{n \to N} \bar{v}_n \to \mu \quad (8)$$

$$\bar{S}_n = \frac{1}{n-1}\sum_{i=1}^{n}(v_i - \bar{v}_n)^2 \quad (9)$$

$$\lim_{n \to N} \bar{S}_n \to \sigma^2 \quad (10)$$

Average Operation

The estimator for the result of average operation by using *n* random samples indicated by $est_n(Q)$ is represented in Equation 11.

$$est_n(Q) = \bar{v}_n \quad (11)$$

Theorem 3.1

$est_N(Q) \approx R(Q)$

The Theorem 3.1 states the effectiveness of $est_N(Q)$ in providing estimates which are closer to $R(Q)$, when the number of samples = *N.* The proof of all the theorems is outlined in Appendix.

Theorem 3.2

The estimator represented in Equation 11 uses $n = \frac{z_p^2 \bar{S}_n}{\epsilon^2}$, *where* z_p *indicates* $\frac{p+1}{2}$ *quantile of the standard normal distribution, then, the sampling condition represented in Equation 4 can be satisfied.*

The Theorem 3.2 gives information about the number of random samples, which need to be extracted to satisfy the estimated quality condition represented in Equation 4.

3.5 Sum Operation

The random samples are transformed through the transformation function represented in Equation 12, which produces another IID observations indicated by $(y_1, y_2, \dots . y_n)$.

$$y_i = Nv_i \; for \; 1 \leq i \leq n \quad (12)$$

The estimator for the result of sum operation is represented in Equation 13.

$$est_n(Q) = \frac{\sum_{i=1}^{n} y_i}{n} \quad (13)$$

Theorem 3.3

$est_N(Q) \approx R(Q)$

Theorem 3.4

The estimator represented in Equation 13 uses $n = \frac{z_p^2 \bar{S}_n}{\epsilon^2}$, *then, the sampling condition represented in Equation 4 can be satisfied. Here,* $\bar{S}_n$ *refers to sample variance of* $(y_1, y_2, \dots . y_n)$.

Variance Operation

The random samples are transformed through the transformation function represented in Equation 14.

$$y_i = (v_i - \bar{v}_n)^2 \; for \; 1 \leq i \leq n \quad (14)$$

The estimator for the result of variance operation is represented in Equation 15.

$$est_n(Q) = \frac{\sum_{i=1}^{n} y_i}{n} \quad (15)$$

Theorem 3.5

$est_N(Q) \approx R(Q)$

Theorem 3.6

The estimator represented in Equation 15 uses $n = \frac{z_p^2 \bar{S}_n}{\epsilon^2}$*, then, the sampling condition represented in Equation 4 can be satisfied. Here,* $\bar{S}_n$ *refers to sample variance of* $(y_1, y_2, \ldots . y_n)$.

Standard Deviation Operation

The random samples are transformed through the transformation function represented in Equation 16.

$$y_i = \sqrt{n}(v_i - \bar{v}_n) \; for \; 1 \leq i \leq n \quad (16)$$

The estimator for the result of Standard Deviation operation is represented in Equation 17.

$$est_n(Q) = \frac{\sum_{i=1}^{n} y_i}{n} \quad (17)$$

Theorem 3.7

$est_N(Q) \approx R(Q)$

Theorem 3.8

The estimator represented in Equation 16 uses $n = \frac{z_p^2 \bar{S}_n}{\epsilon^2}$*, then, the sampling condition represented in Equation 4 can be satisfied. Here,* $\bar{S}_n$ *refers to sample variance of* $(y_1, y_2, \ldots . y_n)$.

Estimation of the Population Size

Let, $[u_1, u_2, u_3 \ldots . u_n]$ represent the number of tuples inside the blocks selected for the corresponding random samples $[v_1, v_2, \ldots v_n]$ respectively. The estimator for *N* indicated by $\widehat{N}_n$ is represented in Equation 18.

$$\widehat{N}_n = \frac{m \sum_{i=1}^{n} u_i}{n} \quad (18)$$

Theorem 3.9

$$\widehat{N}_m = N$$

Theorem 3.9 states that, $\widehat{N}_m$ is an consistent estimator of *N,* and hence, $\widehat{N}_n \approx N$.

Algorithm

The proposed approximation technique for aggregate query is outlined in Algorithm 1. The function $user\_input()$ collects the user quality parameters p and ∈. Each block of the result set is assigned a unique index value through $create\_index\_block(m)$. The required number of samples for approximating Q is calculated through *sample_size()*, which utilizes the result of Theorem 3.2, 3.4, 3.6 or 3.8 depending upon the aggregate operation of *Q*.

Each result set block is assigned to a unique Map Task. Random selection of a Map Task is performed through $random\_select\_map\_task(m)$. The selected Map Task is executed through *Map_Task()* outlined in Algorithm 2, which performs tuple structuring in the corresponding block through *create_index_tuple()*, produces the required random sample through *random_select_tuple()*, and the corresponding block size is calculated. This random sample production continues until the required number of random samples is produced. The reduce task is invoked through *Reduce_Task()* outlined in Algorithm 3*,* which produces required $est_n(Q)$ by invoking *result_estimate()*, which utilizes Equations 11, 13, 15 or 17 depending on the aggregate operation of *Q,* and also Equation 18 is utilized if required.

Algorithm 1: Approximation Technique for Aggregate Query

$[p, \in] = user\_input()$
$create\_index\_block(m)$
if $Q == avg$ **then**
$n = sample\_size(Q, avg)$
end if
if $Q == sum$ **then**
$n = sample\_size(Q, sum)$
end if
if $Q == var$ **then**
$n = sample\_size(Q, var)$
end if
if $Q == SD$ **then**
$n = sample\_size(Q, SD)$
end if
$k = 1$
while $k \leq n$
$map = random\_select\_map\_task(m)$
$[v_k, u_k] = Map\_Task(map)$
$k = k + 1$
end while
$est_n(Q)$=*Reduce_Task(Q,*$\{ [v_k, u_k] \}$*)*

Algorithm 2 *Map_Task(map)*

structure(map)
create_index_tuple(map)
tuple=random_select_tuple(map)
Return [*tuple,block_size]*

Algorithm 3 *Reduce_Task(Q,*{ $[v_k, u_k]$}*)*

If *Q==avg* **then**
$est_n(Q)$=*result_estimate(Q,avg,*{ $[v_k, u_k]$})
end if
If *Q==sum* **then**
$est_n(Q)$=*result_estimate(Q,sum,*{$[v_k, u_k]$})
end if
If *Q==var* **then**
$est_n(Q)$=*result_estimate(Q, var,*{ $[v_k, u_k]$})
end if
If *Q==SD* **then**
$est_n(Q)$=*result_estimate(Q, SD,*{ $[v_k, u_k]$})
end if
Return $est_n(Q)$

RESULTS AND DISCUSSION

System Setup

The proposed approximation technique for aggregate query is implemented through Hadoop framework. The publically available TPC-DS dataset is used for empirical analysis. Even-though, this dataset is structured, structuring labels are removed to create an unstructured version. The test queries are self designed. The values of the different system parameters are outlined in Table 1.

Table1: System Parameter Settings

Parameters	Values
Dataset size	3TB
Number of test queries of each aggregate operation	10
Total number of qualifying tuples in the entire dataset for a single aggregate query.	Varied between 3% to 20% portion of the entire dataset.
Number of data blocks	10^6
P	Varied between 0.7-0.9
∈	30% error w.r.t actual result
Number of computing nodes	10
Number of data blocks assigned to each computing node	10^5

The metric *esratio* represented in Equation 19 is used for empirical analysis. Here, higher values of *esratio* reflect ineffectiveness of the estimator. Similarly, lower values of *esratio* indicate that, the estimator provides approximate results closer to the actual result.

$$esratio = \frac{|est_n(Q) - R(Q)|}{R(Q)} \times 100 \qquad (19)$$

Empirical Result and Discussions

The first experiment evaluates the *esratio* performance and execution time of the proposed approximation technique for aggregate queries w.r.t. average operation. Three different cases are analyzed, wherein, three different values of *p* are respectively utilized. The result of this analysis w.r.t. *esratio* is illustrated in Figure 1. Here, the different test queries are indicated by their query numbers. It is clear that, for higher values of *p* the estimator provides better accuracy mainly due to the confidence requirement of the estimator reflected through *p*.

The execution time performance of the proposed estimator for the first experiment is illustrated in Figure 2. Since, the number of random samples required to meet the estimator accuracy goal is directly proportional to the parameter *p***,** the estimator incurs higher execution cost. However, this increased cost is not in the orders of magnitude scale.

The second, third and fourth experiments analyzes the *esratio* and execution time performance of the proposed approximation scheme for sum, variance and standard deviation respectively. The analysis result of these experiments is illustrated in Figures 3, 4, 5, 6, 7 and 8. Similar results seen in the first experiment are also seen here.

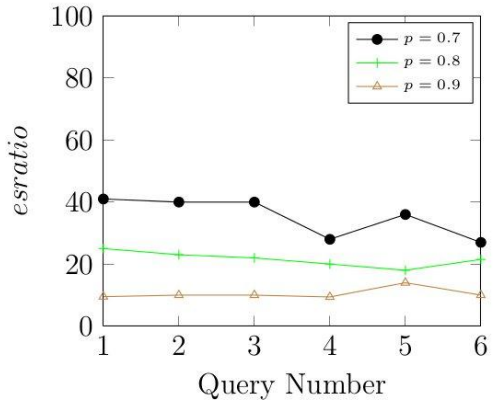

Figure 1: *esratio*(Average)

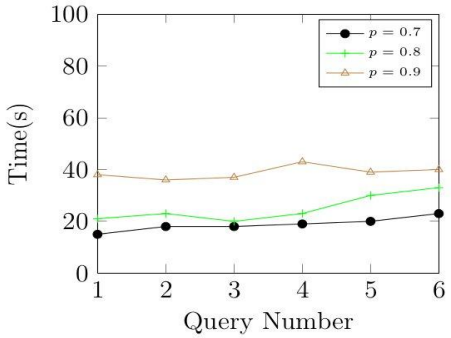

Figure 2: Execution Time(Average)

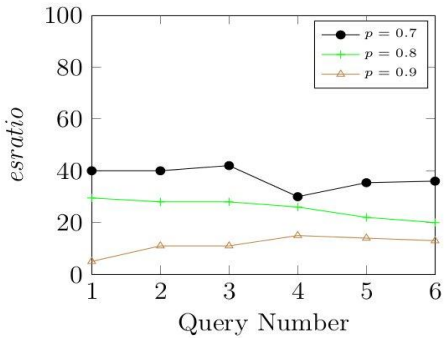

Figure 3: *esratio*(Sum)

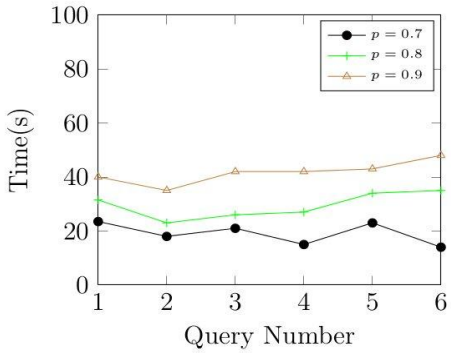

Figure 4: Execution Time(Sum)

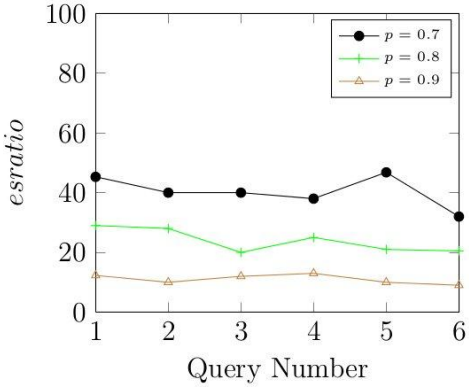

Figure 5: *esratio*(Variance)

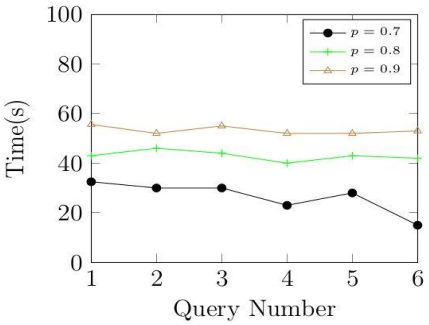

Figure 6: Execution Time(Variance)

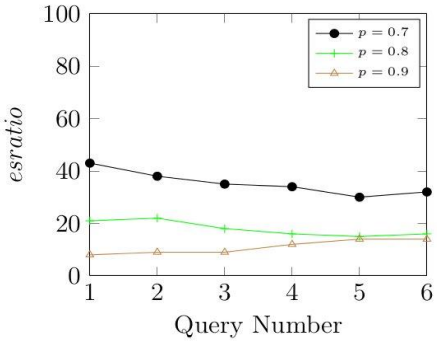

Figure 7: *esratio*(SD)

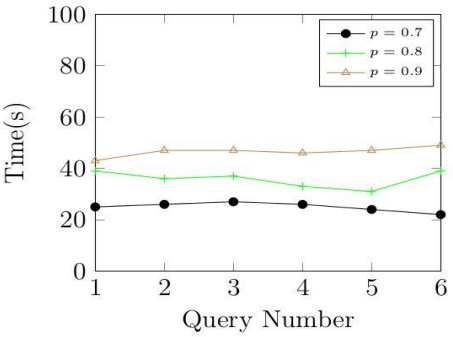

Figure 8: Execution Time(SD)

Conclusion

In this work, the importance and challenges of sampling schemes in evaluating Big Data aggregate queries was summarized. A new approximation scheme for estimating the result of aggregate queries was presented, which overcomes the listed challenges. Four different aggregate operations were addressed—average, sum, variance and standard deviation. The sampling scheme was built through the framework of CLT. Empirical results establish the effectiveness of the proposed scheme by providing estimated results closer to the actual result. Also, the proposed scheme incurs limited execution cost.

The future work is to investigate effective sampling schemes for evaluating distinct values present in the attribute w.r.t. Big Data environment.

APPENDIX

Proof of Theorems

Theorem 3.1

Proof. Since, m is very large, it can be inferred that, $n_1 \approx n_2 \approx \cdots \approx n_m$. This implies that, all the qualifying N tuples are equally likely to be selected, and almost all the N tuples will appear once in the set $(v_1, v_2, \ldots . v_N)$. Hence, the theorem immediately follows.

Theorem 3.2

Proof. Since, $SN(z_p) = \frac{p+1}{2}$, by letting $p = 2SN\left(\frac{\epsilon\sqrt{n}}{\bar{s}_n}\right) - 1$ in Equation 6, and since, $\mu \approx est_N(Q)$, the theorem immediately follows.

Theorem 3.3

Proof. Since, the transformed set $(y_1, y_2, \ldots . y_N)$ is created by multiplying N to the set $(v_1, v_2, \ldots . v_N)$, the probabilities of the transformed set are the same as the original set. The remaining part of the proof is on the same lines outlined in Theorem 3.1.

Theorem 3.4

Proof. In Equation 6, replace $\bar{v}_n$ and μ by $\bar{y}_n$ and μ_y respectively, wherein, $\bar{y}_n$ indicates the sample mean of $(y_1, y_2, \ldots . y_n)$ and μ_y indicates the mean of $(y_1, y_2, \ldots . y_n)$. Since, est_n(Q) is a consistent estimator of μ_y, it can be inferred that, $\mu_y \approx est_N(Q)$. Let, $p = 2SN\left(\frac{\epsilon\sqrt{n}}{\bar{s}_n}\right) - 1$. By using Theorem 3.3 and Equation 6, the theorem immediately follows.

Theorem 3.5

Proof. It is clear from Theorem 3.3 that, $(y_1, y_2, \ldots . y_N)$ is created by considering almost all the qualifying tuples for Q. So, $R(Q) \approx \frac{\sum_{i=1}^{N} y_i}{N-1}$. Since, N is large, $N - 1 \approx N$, and the theorem immediately follows.

Theorem 3.6

Proof. The proof is similar to the proof presented for Theorem 3.4, and by using Theorem 3.5, the theorem immediately follows.

Theorem 3.7

Proof. Here, $R(Q) = \frac{\sqrt{\sum_{i=1}^{N}(v_i - \bar{v}_N)^2}}{\sqrt{N-1}}$. Consider the metric $\hat{R}(Q) = \frac{\sum_{i=1}^{N}(v_i - \bar{v}_N)}{\sqrt{N-1}}$. By using the result $\sqrt{x} + \sqrt{y} \leq \sqrt{x+y}$, where x,y and z are some real numbers, and $\sqrt{x} + \sqrt{y}$ is closer to $\sqrt{x+y}$. It can be inferred that, $\hat{R}(Q) \approx R(Q)$. Since, N is large, it implies that, $N - 1 \approx N$, which implies that, $est_N(Q) \approx \hat{R}(Q)$, and the theorem immediately follows.

Theorem 3.8

Proof. The proof is similar to the proof presented for 3.4. By using Theorem 3.7, the proof immediately follows.

Theorem 3.9

Proof. The term $\frac{\sum_{i=1}^{n} u_i}{n}$ indicates the sample mean for the observations $(u_1, u_2, \ldots . u_n)$. The term m is a constant. By using the fact that, sample mean is a consistent estimator of the actual mean, and due to the strong law of large numbers, the theorem immediately follows.

REFERENCES

[1] J.M.Hellerstein, P.J.Haas and H.J.Wang online aggregation In SIGMOD Conference, 1997,pp.171-182.

[2] C.M.Jermaine, S.Arumugam, A.Pol, and A.Dobra Scalable approximate query processing with the dbo engine In SIGMOD Conference, 2007, pp.725-736.

[3] M.L. Kersten, S.Idroes, S. Manegold, and E.Liarou The researchers guide to the data-deluge:Querying a scientific database in just a few seconds In PVLDB, vol.4, no.12, pp. 1474-1477, 2011.

[4] S.Chaudhari, V.R. Narasayya, and R. Ramamurthy Estimating progress of execution for SQL queries In SIGMOD, 2004.

[5] G.Luo, J.F. Naughton, C.J. Ellman, and M.Watzke Toward a progress indicator for database queries In SIGMOD 2004.

[6] A.Ganapathi, H.A Kuno, U.Dayal, J.L.Weinter, A.Fox, M.I.jordan, and D.A. Patterson, Predicting multiple metrics for queries: Better decisions enabled by machine learning In ICDE 2009.

[7] M.Akdere, U. Cetintemel, M. Riondato, E. upfal, and S.Zdonik Learning based query performance modeling and prediction In ICDE,2012.

[8] Y.E. Ioannidis, The history histograms (abridged)In VLDB, 2003, pp.19-30.

[9] R.J. Lipton, J.F. Naughton and D.A Schneider Practical selectivity estimation through adaptive sampling In SIGMOD, 1990.

[10] P.J.Haas, J.F. Naughton, S.Shehsadri, and A.N.SwamiSelectivity and cost estimation for joins based on random sampling In J.Comput.Syst. Sci., vol.52,no.3, pp.550-569, 1996.

[11] M.Charikar, S.Chaudhuri, R.Motwani, and V.R.Narasayya Towards estimation error guarantees for distinct values In PODS, 2000, pp.268-279.

[12] S.Abiteboul, R.Hull, and V.Vianu, Foundations of Databases In Addison-Wesley, 1995.

[13] S.Ross, A First Course in Probability, 8th ed. Prentice Hall,2009.

[14] R.J.Lipton and J.F.naughton Query Size estimation by adaptive sampling In PODS, 1990.

[15] P.J. Haas and A.N. Swami, Sequential Sampling Procedures for query size estimation In SIGMOD Conference, 1992, pp.341-350.

[16] R.J.Lipton, J.F. Naughton and D.A. Schneider Practical selectivity estimation through adaptive sampling University of Wisconsin-Madison Computer Sciences Department, Tech. Rep., 1990.

[17] S.Chaudhuri, R.Motwani, and V.R. NarasayyaOn random sampling over joins In SIGMOD Conference, 1999, pp.263-274.

[18] J.Bunge and M.Fitzpatrick Estimating the number of species Journal of the American Statistical Association, vol.88, no421, pp.364-373, 1993.

[19] P.J. Haas, J.F. Naughton, S. Seshadri and L.Stokes Sampling-based estimation of the number of distinct values of an attribute In VLDB, 1995, pp.311-322.

[20] Raman Grover, Michael J. Carey Extending Map Reduce for Efficient Predicate Based Sampling In Technical Report.

[21] V.Ayyalasomayajula. HERDER A Heterogeneous Engine for Running Data-Intensive Experiments and Reports M.S. Thesis University of California-Irvine, 2011.

[22] S.Babu Towards automatic optimization of map reduce programs In proc. soCC 2010 pp.137-142.

[23] S.Chaudhuri, R.Motwani and V.Narasayya On random sampling over joins. In Proc. SIG-MOD 1999, pp.263274.

[24] J.Dean and S.Ghemaeat. Map Reduce Simplified data processing on large clusters In Proc OSDI 2004, pp.137-150.

[25] Hadoop. http://hadoop.apache.org.

[26] Hive Website. http://hadoop.apache.org/hive.

[27] https://issues.apache.org/jira/browse/HIVE-2004.

[28] C.Olston, B.Reed et.al. PigLatin A not-so-foreign language for data processing In Proc SIGMOD 2008, pp1099-1110.

[29] A.Thusoo, Z. Shao et al. Data warehousing and analytics infrastructure at Facebook In Proc. SIGMOD 2010, pp.1013-1020.