International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 6, December 2017, pp. 3745~3752
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i6.pp3745-3752

Evaluating Aggregate Functions of Iceberg Query Using Priority Based Bitmap Indexing Strategy

Kale Sarika Prakash[1], P. M. Joe Prathap[2]
[1,2]Departement of Computer science and Engineering, St. Peter's Institute of Higher Education and Research, St. Peter's University, Avadi Chennai, India

Article Info

Article history:

Received Apr 10, 2017
Revised Sep 8, 2017
Accepted Sep 29, 2017

Keyword:

Iceberg query (IBQ)
Bitmap index (BI)
Aggregate functions
Logical operations
Data warehouse (DW)

ABSTRACT

Aggregate function and iceberg queries are important and common in many applications of data warehouse because users are generally interested in looking for variance or unusual patterns. Normally, the nature of the queries to be executed on data warehouse are the queries with aggregate function followed by having clause, these type of queries are known as iceberg query. Especially to have efficient techniques for processing aggregate function of iceberg query is very important because their processing cost is much higher than that of the other basic relational operations such as SELECT and PROJECT. Presently available iceberg query processing techniques faces the problem of empty bitwise AND,OR XOR operation and requires more I/O access and time.To overcome these problems proposed research provides efficient algorithm to execute iceberg queries using priority based bitmap indexing strategy. Priority based approach consider bitmap vector to be executed as per the priority.Intermediate results are evaluated to find probability of result.Fruitless operations are identified and skipped in advance which help to reduce I/O access and time.Time and iteration required to process query is reduced [45-50] % compare to previous strategy. Experimental result proves the superiority of priorty based approach compare to previous bitmap processing approach.

Corresponding Author:

Kale Sarika Prakash,
Departement of Computer science and engineering,
St.Peter's Institute of Higher Education and Research,
St. Peter's University, Avadi, Chennai, India.
Email: kalesarikaprakash@gmail.com

1. INTRODUCTION

Data warehouse (DW) is collection of subject oriented, integrated, non-volatile and time variant dataset [1]. Analysis of data from data warehouse is very important factor for the decision making in any business organization.As data warehouse is huge so analysis is also complex because for analysis multidiensionl approach is required [2].Analysis of such huge database is done by executing complex queries such as iceberg query (IBQ) and online Anylytical processing functions.The basic operation required in DW analysis is aggregate functions such as MIN,MAX,SUM,AVG and COUNT. Generally the queries to be executed on DW are the queries with aggregate function followed by HAVING and GROUP By clause, such a query is known as IBQ. It consists of three main parameters such as aggregate function, HAVING clause and GROUP BY clause which makes the query more complex.

In addition to the complexity of IBQ, the large volume of data stored in DW lengthens the time needed to run queries. Hence performance of query in terms of time is most important requirement of any large database system. This research focus on efficient execution of aggregate function as it is main part of IBQ. Aggregates function can have a significant impact on performance of query in term of time.For efficient execution of aggregate function we require efficient and fast processing of huge data [3].

The main objective of this research is efficient execution of aggregate function of IBQ. To achieve this we are making use of priority based bitmap indexing (BI) strategy. Number of researchers [4]-[8] work to improve performance of IBQ. But all of them faces the problem of empty bitwise AND, OR XOR operation as well as futile queue pushing problem. This research overcomes these problems by using priority based BI strategy. This strategy first analyse which operation to be perform as per the evaluation of query .According to evaluation process it arrange the sequence of operations to be perform. Based on results of current operation it change the priority and perform the remaining operation. In this way in between this technique identify useless operation in advance and skip such useless operations. Thus by performing only required operations it reduces I/O access as well as time required to execute IBQ. This strategy work on bitmap vector of attribute as per query requirement. The Bitmap vectors are in the form of 0's and 1's and proposed strategy perform logical operations such as OR, AND and XOR on this bitmap vectors. Executing bitwise operations on 0's and 1's are very much cost effective in term of I/O access and time. It directly helps to improve IBQ performance. Our experimental result proves that performance of our strategy is better than previous algorithms.In future by extending this concept on unstructured data it will be applicable for big data analysis [9].

2. REVIEW OF BI, AGGREGATE FUNCTION AND IBQ PROCESSING METHOD

Bitmap indexing technique is most suitable and efficient for read mostly, append only data and large size dataset.BI is commonly used in the DW application. BI strategy performs better than tree based indexing methods like different type of B Tree and R Tree [10]. BI has two advantages for using it in DW are it avoids complete table scan and saves disk access [11],[12]. This research makes use of compressed BI concept which saves the memory and shows the effectiveness of BI for IBQ evaluation [4]. BI performs effectively as it works on index level rather on original table. This feature help to improve performance in terms of time required to execute query, memory required to store database and I/O access cost.By considering all above features of BI we are using it in our research.

Aggregation functions across many attributes are commonly used in queries of data mining, DW and OLAP [13],[14]. The commonly used queries in data mining and DW are IBQ, which perform an aggregate function across attributes and then remove aggregate values that are below some specified threshold value. Generally used aggregation functions are MIN,MAX,SUM,AVG and COUNT. Efficient computation of all these aggregate functions is required in most large database applications because processing cost of aggregate function is much higher than that of the other basic relational operations like SELECT and PROJECT.

IBQ refer to a class of queries which compute aggregate functions across attributes to find aggregate values above some specified threshold value. The number of tuples, that satisfy the threshold in the having clause, is relatively small compared to the large amount of input data. As output result is very small so time required for extracting it must be less.Syntax of IBQ is as below.Given a relation R with attributes a_1, a_2… a_n , an aggregate function AggFun(), and a threshold T.

SELECT a_1, a_2… a_n, AggFun(*)
FROM relation R
GROUPBY a_1, a_2… a_n
HAVING AggFun (*) >= T

IBQ concept is first studied by Min Fang[10] in 1998.In this research researchers extend probabilistic technique used in [15] and proposes hybrid and multi bucket algorithm .This research combine sampling and multi hash functions to improve the performance of IBQ and reduce memory requirement. But these algorithms are not suitable for large data sets. To solve above problem [10] proposes algorithms based on sampling and bucket counting methods. These methods reduces number of false positive values but it takes more time to execute query as it require multiple scan of relation.

IBQ processing is also proposed by [16], focus of this study is to reduce number of table scans so that time required to execute the query will get reduced. It introduces methods to select candidate values using partitioning and postpone partitioning algorithms.

Collective IBQ Evaluation is proposed by [18] which present comparison using three methods sort merge aggregate, hybrid hash aggregate and ORACLE. This study proves that performance of sort merge aggregate is better on data sets with low to moderate number. Hybrid hash aggregate performance was not good when data set is large. All above mentioned methods comes under the group of tuple scan based, which requires one physical table scan to read data from disk.However [18] tries to make use of this property of IBQ and uses BI but it suffers from empty bit wise AND result problem. Researchers [4] tries to minimize this problem using dynamic pruning and vector alignment algorithms .However they notice that there is problem of massively empty bitwise AND results and extra XOR operation. To overcome this challenge they

develop vector alignment algorithm which help to solve empty bitwise AND operation problem. The problem with this algorithm is that all vectors may not have 1 bit at same position and if it is not at same position then all the AND as well as XOR operations are fruitless and time consuming.In this way both the above approaches[4] suffer from fruitless AND as well as XOR operations. Research [5] try to handle empty XOR operation problem but did not able to solve fruitless bit wise AND operation problem. Both the research [4] and [5] faces the problem of futile queue pushing.

In this paper, we have solved the problem of fruitless bitwise AND, OR and XOR operation and futile queue pushing by using priority based BI strategy. This approach improves efficiency by pruning many groups beforehand. This research used the similarity matching concept before assigning priority to vector and forms the cluster of the same [19].

3. PRIORITY BASED BI STRATEGY FOR IBQ EVALUATION

3.1. Working model of Priority based BI strategy for IBQ evaluation

This section describes the workflow of priority based BI strategy for IBQ evaluation. As shown in Figure 1 priority based approach is work along with tracking pointer strategy as well as look ahead matching method. Once the bitmap vector is generated then priority based approach will make use of tracking pointer concept to assign priority to vectors as per the position of 1's occur in vector. After finalization of vectors for performing bitwise AND operation then the look ahead matching strategy will get activate to find out probability of that operation whether it will satisfy threshold condition or not. If it recognize that possibility of success is less then it will skip further AND operation .In this way it help to reduce unnecessary burden of performing fruitless bitwise AND operation. Finally our module 5 will execute compare operation to combine all the result which satisfies threshold condition.

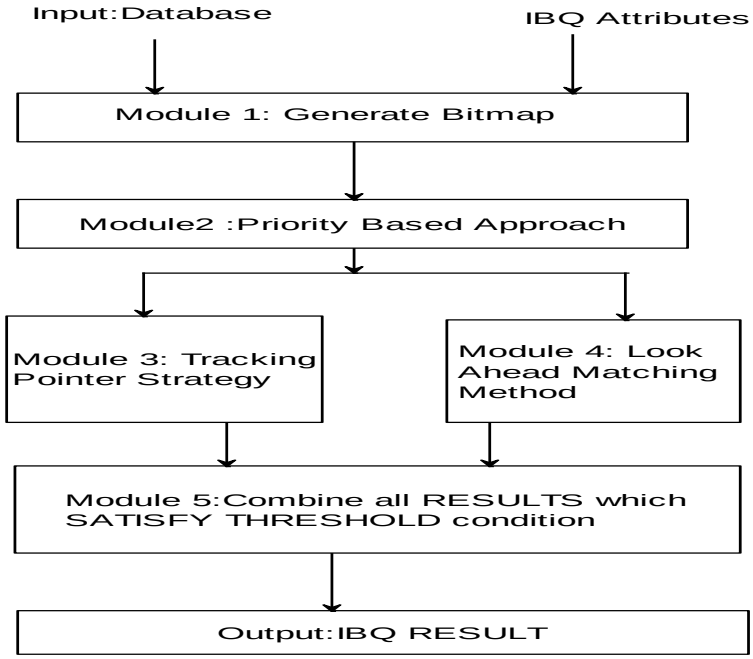

Figure 1. Workflow Diagram

3.2. Pseudo code for Priority based BI strategy for IBQ evaluation

This subsection represents the processing flow of priority based BI strategy for IBQ evaluation. This strategy is mainly work in three phases like generating BI, tracking pointer strategy and look ahead matching method.The work flow of algorithm is as below:

Input: (Iceberg Query(Attribute X, Attribute Y, threshold T), Table P, Bitmap Vector table of P)
Processing: Processing of algorithm is based on number of distinct values of IBQ attribute and Threshold,
Output: (IBQ RESULT)

Phase 1: Create Bitmap Vector Generation Function

It contains main functions which are used to convert INPUT into OUTPUT. First Function is Create BTMAP VECTOR on IBQ attribute. It works on following formula:

$$\text{BITMAP VECTOR} = [(\text{cardinality of Cloumn A} + \text{cardinality of Cloumn B} + \cdots \ldots + \text{cardinality of Cloumn N})] \times \text{No. of Rows present in Database}]$$

This formula also used to find the Space Complexity of Algorithm. Relationship between each cardinality is one to one means one vector related to one only. The attribute which has this relationship is SET to 1 otherwise 0. In this way complete BITMAP VECTOR is created.

Phase 2: Tracking Pointer Strategy

1. For each bitmap vector a of Attribute X COUNT (Number of 1's in each Bitmap vector) if it is > T then only keep such vector in BI. Otherwise discard it from the list. For each bitmap vector a of Attribute X. Find first 1 bit position and accordingly allocate Priority Priority Queue X. clear, Priority Queue Y. clear. For each vector x of attribute X do
   If(x.count>=T)then x.next1=FirstOneBitPosition(x,0)
2. For each bitmap vector a of Attribute Y
   COUNT (Number of 1's in each Bitmap vector) if it is > T then only keep such vector in BI otherwise discard it from the list.
   Priority Queue X. clear, Priority Queue Y. clear. For each vector y of attribute Y do
   If(y.count>=T)then y.next1=FirstOneBitPosition(y,0)
3. Find first 1 bit position of vector X and Y and accordingly allocate Priority.
   If (X.Positionof1Bit > Y. Positionof1Bit)
   Then (FirstPriority == X.vector)
   Else (FirstPriority == Y.vector)
4. If (X.Positionof1Bit == Y. Positionof1Bit)
   Then (FirstPriority == X.vector) as X vector appears first in sequence and Y comes later.
5. PriorityQueueX.Push(x)
6. PriorityQueueY.Push(y)
7. x,y=NextMatchVector(PriorityQueueX.clear, PriorityQueueY,T)
8. While x!=NULL &y!=NULL do
9. PriorityQueueX.Pop
10. PriorityQueueY.Pop
11. CurrentResult=BitwiseAND(x,y)
12. If(CurrentResult.count>=T) then
13. Add IBQ Result in RESULT(x.value,y.value,CurrentResult.count)
14. x.count=x.count-CurrentResult.count
15. y.count=y.count-CurrentResult.count
16. If x.count>=T then
17. x.next1=FirstOneBitPosition(x,x.next+1)
18. If x.next1!=NULL then
19. PriorityQueueX.Push(x)
20. If y.count>=T then
21. y.next1=FirstOneBitPosition(y,y.next+1)
22. If y.next1!=NULL then
23. PriorityQueueY.Push(y)
24. Repat step 7-23 for next vector
    x,y=NextMatchVector(PriorityQueueX, PriorityQueueY,T)

Phase 3: Look ahead matching method

If RESULT satisfies THRESHOLD condition then to predict the possibility of fruitful result look ahead matching strategy is used. This help to reduce fruitless AND,OR and XOR operation. It prune the vector as it identify that this vector will not able to produce positive result .In this way this module skip further operational overhead of IBQ processing.

25. GENERATE new vectors by performing OR operation between RESULT and the new vector which is already part of RESULT.
    New X Vector = Old X vector- CurrentResult Vector

New Y Vector = Old Y vector- CurrentResult Vector

26. If (New X/Y Vector) satisfy Threshold condition then perform step 7-23 on newly generated vector otherwise skip the respective attribute from the vector list.
    This step helps to identify the possibility of vector to be part of RESULT further.
27. Repeat step 7-26 till the vector list will be empty.
28. Return IBQ RESULT

4. RESULTS AND DISCUSSION

The proposed method is implemented using JAVA platform on the IBM-compatible PC with Intel(R) Core i3 processor @ 3.40GHz and 2GB RAM. The experiment is performed on synthetic dataset of size 5K, 10K, 20K, 40K and 80K. Parameters consider for comparison and to measure the performance are database size, threshold value, number of iterations required to execute query, time and aggregate functions. The graphical illustration is shown for COUNT and SUM aggregate functions in Figure 2, 3, 4, 5, 6 and 7.We observed significant improvement in IBQ performance in terms of number of iterations and time required to execute IBQ using our priority based approach (PBA).

We have compare the performance of priority based approach (PBA) with the bitmap indexing approach (BIA) suggested in previous work [4]-[8] .We observe that as we go on increasing size of data set and threshold value then also query performance is goes on increasing which is shown in Figure 2, 3,4 and 5.With previous approache we noticed that as data size increases the time required to extract data is also increases. Based on our experimental result we have proved that through our approach even though data size increases then also IBQ response time get reduced. We are using BI strategy which help to handle huge data effectively [13],[14]. This is also noticed through our experimentation as data size is go on increasing the percentage of response time is reduced .As shown in Figure 3 and 5 through time analysis we observe that for small data set size i.e.5k, 10k and upto 20k difference in time required is only 10-20% but as we go on increasing dataset size from 20k, 40k to 80k difference in time required is reaches to 45-50%. Figure 6 and Figure 7 shows the comparative analysis for iteration and time for SUM function.The number of iterations required are drastically decreases as shown in Figure 7. But due to large database access time requied to execute is reduced to 45-50% only which is as shown in Figure 6. This indicates that our strategy is well suitable for large data set. Through our experimental result we have proved that priority based approach for IBQ processing is superior to the previous bitmap indexing approach. In this way we have developed the frame work for COUNT, SUM, MIN and MAX aggregate function used in IBQ.

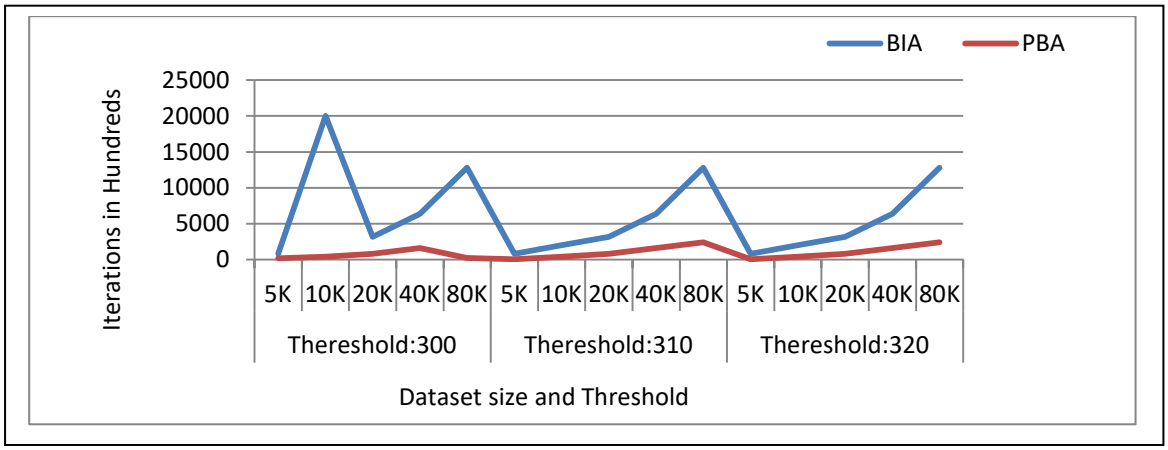

Figure 2. Iteration Analysis of COUNT function

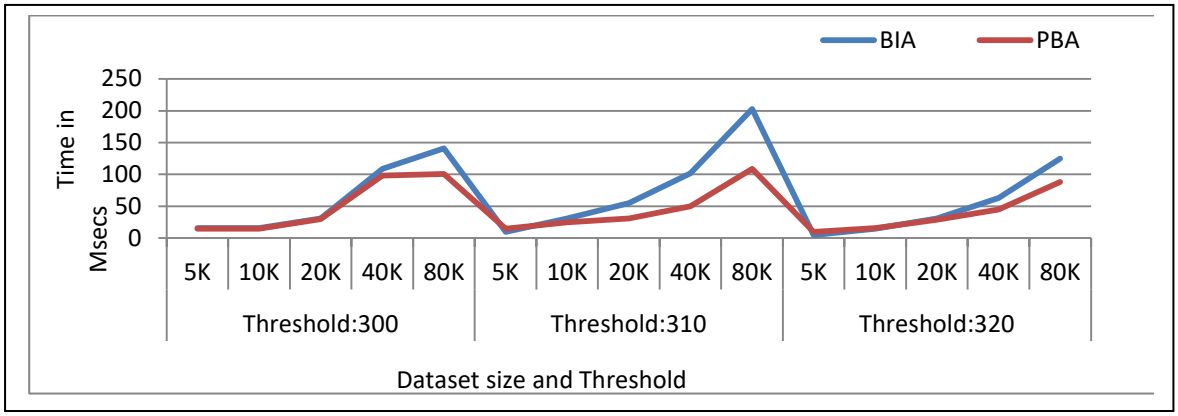

Figure 3. Time Analysis of COUNT function

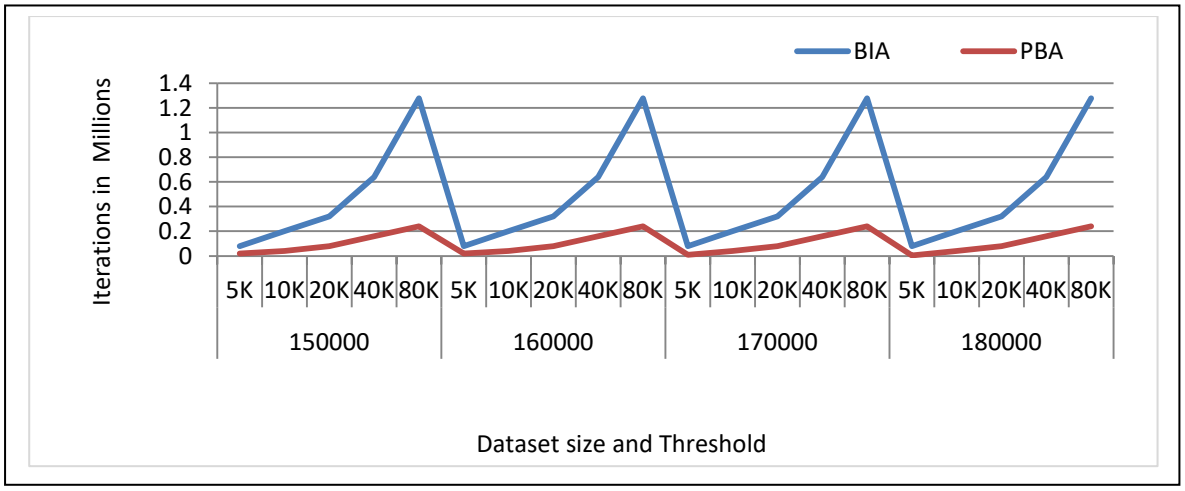

Figure 4. Iteration Analysis of SUM function

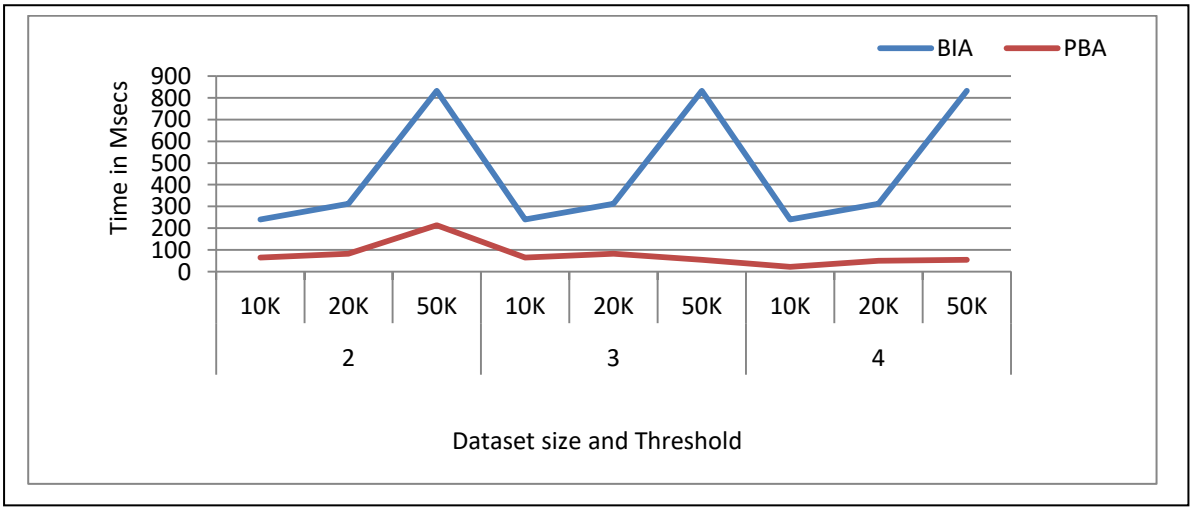

Figure 5. Time Analysis of SUM function

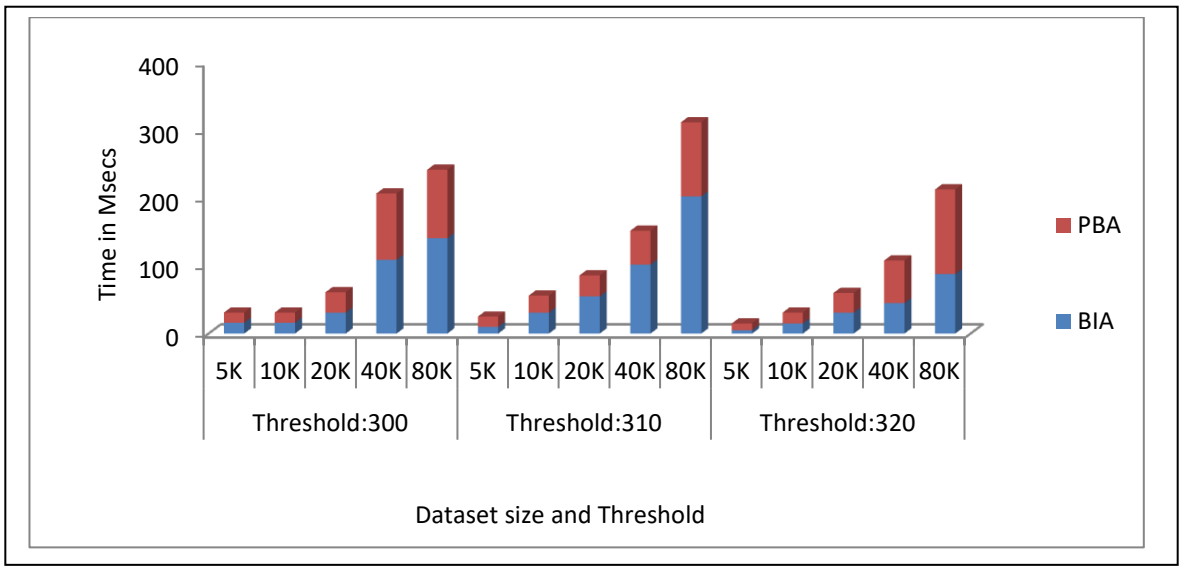

Figure 6. Combine Time Analysis of SUM function

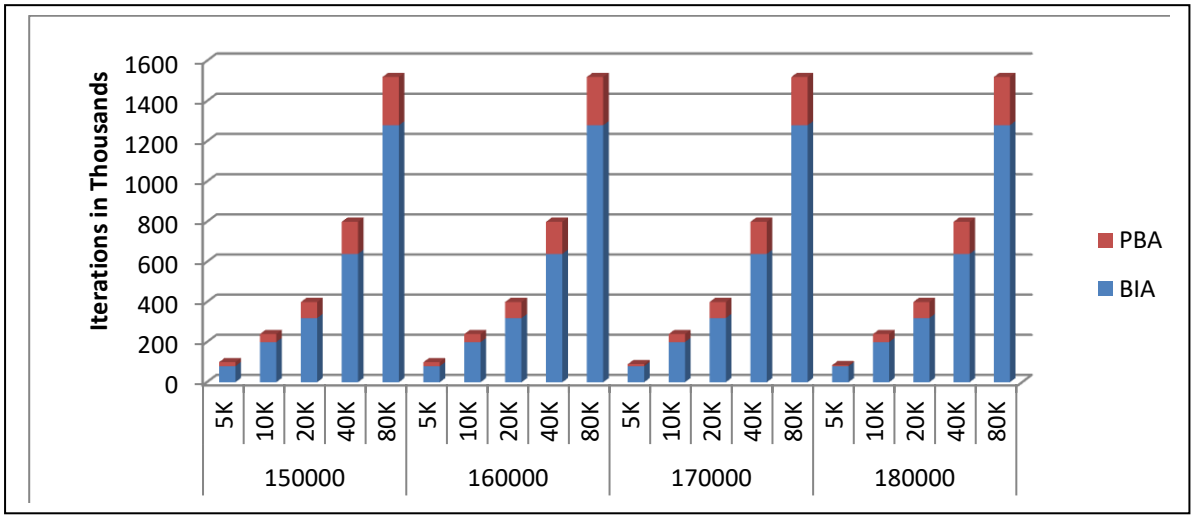

Figure 7. Combine Iteration Analysis of SUM function

5. CONCLUSION

Aggregate functions are the main part of any data analysis task. To analyze the huge dataset like DW we need to execute queries which consist of aggregate function. In such a situation if query is able to execute aggregate function efficiently then it will directly reflect on the performance of query. Intention of this research is to improve efficiency of aggregate functions and IBQ which generally execute on huge data set. In our experimental analysis we compare the performance of our approach with previous work and we notice significant improvement in IBQ performance by using our priority based BI strategy. We noticed that even though the dataset size and threshold value increases then also the data extraction time get reduced. On the basis of experimental result we have proved the superiority of our research.The result of this research will help to execute queries with aggregate function as well as IBQ which improve the performance of OLAP queries on DW. The focus of this research is only structured database but in future we can apply the same logic for query processing on unstructured data and it will helpful for big data analysis.

REFERENCES

[1] W. H. Inmon, "Building the data warehouse," Wiley.com, 2005.

[2] S. Susana, "Query optimization using fuzzy logic in integrated database," *Indonesian Journal of Electrical Engineering and computer science*, vol/issue: 4(3), pp. 637~642, 2016.
[3] A. Dubey, *et al.*, "Effects of Aggregation and Data Size on Query Performance and Memory Requirements of a Data Warehouse," *ICROIT*, 2014.
[4] B. He, *et al.*, "Efficient Iceberg Query Evaluation Using Compressed Bitmap Index," *IEEE Transactions on Knowledge and Data Engineering*, vol/isue: 24(9), pp. 1570-1589, 2011.
[5] C. V. G. Rao and V. Shankar, "Efficient Iceberg Query Evaluation Using Compressed Bitmap Index by Deferring Bitwise- XOR Operations," *IEEE*, 2012.
[6] C. V. G. Rao and V. Shankar, "Computing Iceberg Queries Efficiently Using Bitmap Index Positions," *ICHCI-IEEE*, 2013.
[7] S. Vuppu and C. V. G. Rao, "Cache Based Evaluation of Iceberg Queries," *IEEE International conference on Computer and communication Technologies (ICCCT)*, 2014.
[8] V. C. S. Rao, "Efficient iceberg query evaluation using set representation," *IEEE INDICON*, pp. 1-5, 2014.
[9] M. J. Bashal and K. P. Kaliyamurthie, "An improved similarity matching based clustering framework for short and sentence level text," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 551~558, 2017.
[10] M. Fang, *et al.*, "Computing iceberg queries efficiently," *VLDB Conference*., pp. 299-310, 1998.
[11] J. Gray, *et al.*, "Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals.," *Data Mining and Knowledge Discovery*, pp. 29-53, 1997.
[12] M. Jrgens, "Tree Based Indexes versus Bitmap Indexes: APerformance Study," *Proc. Int'l Workshop Design and Managementof Data Warehouses (DMDW)*, 1999.
[13] An Oracle White Paper, "Oracle Database 11g for Data Warehousing and Business Intelligence," *Oracle*, 2011.
[14] An Oracle White Paper, "Oracle Database 12c-Built for Data warehouse," *Oracle*, 2014.
[15] K. Y. Whang, *et al.*, "A Linear-Time Probabilistic Counting Algorithm for Database Applications," *ACM Trans. Database Systems*, vol/issue: 15(2), pp. 208-229, 1990.
[16] J. Bae and S. Lee, "Partitioning Algorithms for the Computation of Average Iceberg Queries," *Proc. Second Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK)*, 2000.
[17] K. P. Leela, *et al.*, "On Incorporating Iceberg Queries in Query Processors," *Proc. Int'l Conf. Database Systems for Advances Applications (DASFAA)*, pp. 431-442, 2004.
[18] A. Ferro, *et al.*, "BitCube: A Bottom-Up Cubing Engineering," *Proc. Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK)*, pp. 189-203, 2009.
[19] M. Erritali, *et al.*, "An approach of semantic similarity measure between documents based on big data," *International Journal of Electrical and Computer Engineering*, vol/issue: 6(5), pp. 2454~2461, 2016.

BIOGRAPHIES OF AUTHORS

Ms. Kale Sarika Prakash is the research scholor in the department of computer science and engineering at St.Peters University Chennai. She obtained her B.E. (Computer Engineering) from University of Pune, Maharashtra in the year 2000 and M.E. (Computer science and Engineering) from SRTM University, Nanded, Maharashtra in the year 2005.She has been in the teaching profession from the past 17 years. Her area of interest include data mining, data warehousing, big data, business analytics, machine learning,operating system, system programming, software engineering and software testing. She has published 14 papers in various International Journals and Conferences. She has attended many workshops, seminars and FDPs sponsored by ISTE, AICTE and Pune university related to her area of interest. She is a life member of CSI, ISTE & IAENG.

Dr. Joe Prathap P M**,** is an Associate Professor in the Department of Information Technology, since June 2011. He obtained his B.E (CSE) from St. Xavier's Catholic College of Engineering, Chunkankadai, M.E (CSE) from Karunya Institute of Technology, Coimbatore and Ph.D.degree from Anna University, Chennai. He has been in the teaching profession for the past 10 years and has handled both UG and PG programmes. His areas of interest include data mining, machine learning, Computer Networks, Network Security, Operating Systems, Mobile Communication and Object Oriented Analysis and Design. He has published 23 papers in various International Journals and Conferences. He has attended many workshops & FDPs sponsored by AICTE, DST & IEEE related to his area of interest. He is a life member of ISTE & IAENG.