IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Computing SUM and COUNT aggregate functions of Iceberg query using LAM strategy

S.N. Zaware-Kale

Department of Computer Engineering, AISSMS IOIT, SPPU, Pune, India

Corresponding author: sarikazaware@gmail.com

Available online at: www.ijcseonline.org

Accepted: 13/Jan/2019, Published: 31/Jan/2019

***Abstract*—** Aggregate function plays very important role in analyzing data of data warehouse. Analysis of such a huge data requires execution of complex queries such as iceberg and OLAP queries which consist of aggregate function. Improving the performance of such a complex query is the challenge in front of the researchers .Presently available iceberg query processing techniques faces the problem of empty bitwise operations, futile queue pushing and require more table scans. The model proposed in this research applies concept of look ahead matching on bitmap index of query attributes. Based on the threshold value the analysis of logical operation is done in advance. If result satisfies threshold condition then only remaining part will be evaluated otherwise it will be prune and declare as fruitless operation. In this way look ahead matching strategy overcome the problem of previous research. This research proposes framework for SUM and COUNT aggregate function.

***Keywords*—** Aggregate functions(MIN, MAX, SUM, COUNT); Bitwise operations (AND,OR,XOR); Data warehouse(DW); Iceberg query (IBQ); Look Ahead Matching(LAM) strategy

I. INTRODUCTION

DW has emerged as a distinct discipline in the field of information technology. It is used for online analytical processing system(OLAP).DW is a subject oriented, integrated, non-volatile and time variant collection of data which support in management's decisions making process[1]. In OLAP systems, analysis is done by executing different type of queries which run on huge amount of data present in DW [2].The nature of the query to be execute on DW is aggregate function followed by HAVING and GROUP BY clause. Such a type of queries are called as IBQ. IBQ perform an aggregate function across attributes and then eliminate aggregate values that are below some specified threshold. Iceberg queries are so called because the number of above threshold results are often very small that is the tip of an iceberg relative to the large amount of input data .Iceberg queries are also common in many other applications including information retrieval, clustering, and copy detection[3] .

The main part of any IBQ is aggregate function like MIN,MAX,SUM,AVG and COUNT . Hence, the performance of query is depend upon the time required to execute aggregate function. Building aggregates on huge data set and executing IBQ efficiently is challenge in front of current researchers. [4].

This proposed research concentrate on efficient execution of aggregate function and IBQ using bitmap indexing technique. Bitmap Index of required attributes for query processing is in the form of 0's and 1's. Due to this we are performing logical AND,OR and XOR operations as per query requirement. These logical operations can be executed quickly by hardware. The major cost in query processing is the I/O cost as query is going to execute on large database. Due to highly compressible nature, BI have a low I/O cost, and more data can be stored in the main memory for faster query processing.[5].The execution cost of logical operations are cheap which directly help in our strategy to improve the performance of IBQ.

The researchers [4,5,6,7,8]work on improving the performance of IBQ. But they faces the problem of empty bitwise AND operation, empty bitwise XOR operation and futile queue pushing .Proposed research overcome the problem faced by using LAMS during computation of aggregate function and evaluating IBQ. LAM strategy keep track on probability of pruning the vector from further computation in advance. For this it check intermediate results with threshold condition and take decision related to further processing of query evaluation. This minimizes fruitless bitwise AND,OR and XOR operation which reduces query execution time. Experimental result shows the superiority of IBQ evaluation strategy proposed by this research.

This paper is organized as follows. In section II we do a review of aggregate functions and IBQ processing, where we have introduced the concepts of aggregate functions and iceberg queries. In Section III we introduced the concept of LAMS for IBQ evaluation, its pseudo code and performance analysis of LAMS with old method. Section IV is the concluding section of this paper.

II. REVIEW OF AGGREGATE FUNCTIONS AND IBQ PROCESSING

Generally, analysts are interested in summarizing data to determine trends related to their business. This summarized information is useful for top level management of organization for decision making process. For example, the purchasing manager may not be interested in a listing of all computer hardware sales, but may simply want to know the number of laptop sold in a particular month. In such a situations Aggregate functions can assist with the summarization of large volumes of data. An aggregate function F is distributive if there is a function G such that F (T) = G ({F (Si)| i = 1 . . . n}). SUM, MIN, and MAX are distributive with G = F. Count function is distributive with G = SUM.

Efficient computation of all these aggregate functions are required in most of the database applications. The efficient processing of aggregate function on large database is very important because the base for all OLAP operation is aggregate function. If aggregate function is efficient then only query to be execute on large database will perform efficiently.

Here we are considering large size database which is used for analysis of business. Decision related to business can be taken based on previous data of business. Such type of huge and historical data set is known as DW.IBQ and OLAP queries are generally executing on DW where aggregate function computation is needed.

Iceberg queries refer to a class of queries which compute aggregate functions across attributes to find aggregate values above some specified threshold. Given a relation R with attributes a1, a2… an, and m, an aggregate function AggF, and a threshold T, an iceberg query has the form of follow:

SELECT R.att1, R.att2… R.att n, AggFun (F)

FROM relation R

GROUPBY R.att1, R.att2… R.att n

HAVING AggFun Condition (C) >= T

The number of tuples, that satisfy the threshold in the having clause, is relatively small compared to the large amount of input data. The output result can be seen as the tip of iceberg, where the input data is the iceberg..This IBQ processing is first described by Min Fang[3] in 1998.Before this probabilistic techniques were used to process queries with aggregate functions are discussed by K. Whang[10].In [3]author proposed Hybrid and Multi bucket algorithm by extending probabilistic technique used in [10].This research combine sampling and multi hash function concept to improve the performance of IBQ and memory requirement. But these algorithms are not suitable for large size data sets.

To overcome above problem [3] suggest technique which uses combination of sampling and bucket counting mechanism. 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 [11][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. This overcome the problem of multiple scan over relation occurs in sampling and bucket counting mechanism [3].The result of this study shows that performance of above algorithms are degrade due to data order and memory size. If database is sorted then performance is excellent without regards to memory size.

A comparison was presented for Collective Iceberg Query Evaluation (CIQE) [12] using three standard methods like Sort Merge Aggregate (SMA), Hybrid Hash Aggregate (HHA) and ORACLE. Each of these algorithms are having some advantages and disadvantage against each other but common problem with all these algorithm is they come under the group of tuple scan based approach. Execution of these algorithm require one physical table scan to read data from disk. All above algorithms only concentrate on how to reduce the number of table scan none of them make use of properties of IBQ to solve this problem.

To overcome above problem Bitmap index is usually a better choice for querying the massive, high-dimensional scientific datasets. It supports fastest data accesses and reduced the query response time on both high-and low-cardinality values with a number of techniques [13].Generating the bit map index of attribute will not affect on the performance of query because generated bitmap by database system is in compressed mode[14].Therefore use of bitmap index to execute iceberg query avoids the complete table scan.

In [15] researchers tries to make use of IBQ property as well as bitmap index but it suffers from empty bit wise AND result problem. This problem is minimized by[4] using dynamic pruning and vector alignment approaches .This work leverages the antimonotone property of iceberg query and develop dynamic pruning algorithm using bitmap indexing. However they notice that there is problem of massively empty bitwise AND results. To overcome this challenge they develop vector alignment algorithm which uses priority queue concept. The problem with this technique 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 suffer from fruitless AND as well as XOR operations.

Research [6] try to handle empty XOR operation problem but did not able to solve fruitless bit wise AND operation problem. Both the research [4] and [6] uses priority queue concept for all vectors and faces the problem of futile queue pushing.

In proposed research by making use of LAMS we are trying to minimize the number of fruitless bitwise XOR &

AND operation .It will help to minimize futile queue pushing problem and improve the performance of queries with aggregate function.

All of these research [4],[6],[7],[8] and [9] work on COUNT aggregate function only . In proposed research we are developing framework for aggregate functions like SUM and COUNT.

III. WORK FLOW OF LAMS FOR IBQ PROCESSING

A. Fundamental Strategy

Suppose, a purchase manager is given a sales transaction dataset, he/she may want to know the total number of products, which are above a certain threshold T, of every type of product in each local store. To answer this question, we can use the iceberg query below:

```
SELECT    Product_Type, Location,  Sum ( Product)
FROM      Relation Sales_report
GROUPBY   Product_Type ,Location
HAVING    Sum ( Product) >= Threshold
```

To implement iceberg query, a common strategy in horizontal database is first to apply hashing or sorting to all the data in the dataset, then to count all of the location & Product Type pair groups, and finally to eliminate those groups which do not pass the threshold T. But these algorithms can generate significant I/O for intermediate results and require large amounts of main memory. They leave much room for improvement in efficiency. One method is to prune groups using the Apriori-like[16] method. But the Apriori-like method is not always simple to use for all the aggregate functions. For instance, the Apriori-like method is efficient only in case of SUM aggregate function[17]. In our method, instead of counting the number of tuples in every location & Product Type pair group at first step, we can do the following: Generate Location-list: a list of local stores which sell more than T number of products. For example,

```
SELECT    Location, Sum ( Product)
FROM      Relation Sales_report
GROUP BY  Location
HAVING    Sum ( Product) >= Threshold
```

Generate Product Type-list: a list of categories which sell more than T number of products.

For example,

```
SELECT    Product_Type, Sum ( Product)
FROM      Relation Sales
GROUPBY   Product _Type
HAVING    Sum ( Product) >= Threshold
```

From the knowledge we generated above, we can eliminate many of the location & Product _Type pair groups. It means that we only generate candidate location and Product Type pairs for local store and Product type which are in Location-list and Product Type-list. This approach improves efficiency by pruning many groups beforehand. Figure 1 shows the step wise working model of LAM Strategy.

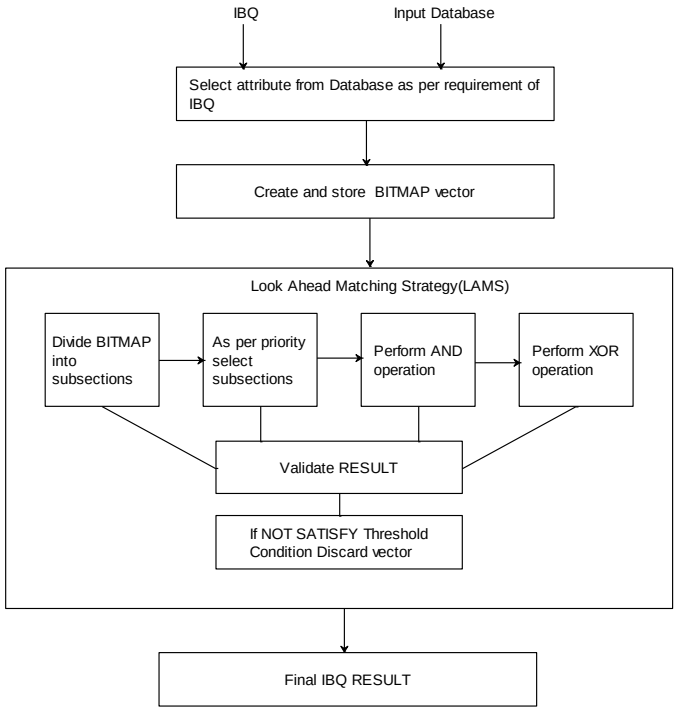

Figure 1: Working model of LAMS

LAM strategy is used during bitwise AND operation. The work flow of this strategy is based on dividing the Input vectors into subparts. Perform the operation on subpart and go on adding the results of other parts. The advantage of this strategy is that in between if we notice that our result cross the Threshold value then there is no need to perform remaining bit wise operation. In this way in advance we are analyzing results and avoid unnecessary operations. In this way this strategy helps to improve query performance by reducing number of I/O access and processing time.

B. Pseduo Code of look ahead matching strategy

Input to algorithm is (Iceberg Query, Input Database), Iceberg Query parameters are (attribute Location , attribute Product, threshold T) and output contain combination which satisfy IBQ threshold condition(T).

Algorithm:

- PriorityQueueA. clear, PriorityQueueB.clear for Location and Product bitmap vector
- Divide the input vector into subparts.P=(L/T) where L=length of vector, T=Threshold value .P=Number of subparts={P1,P2,...,Pn}.
- Start with subpart P1 of each attribute vector Location and Product do
- Perform Bitwise AND operation
- If Bitwise_AND_Result>T
- Then Skip Bitwise AND operation of remaining subpart and send the current combination for XOR operation
- Else store Bitwise_AND_Result and Perform operation on next subpart. After every operation go on checking Result with Threshold.
- If all subparts are finished and Bitwise_AND_Result is not > T then simply discard the combination from the list.In this way pruning the vector which does not satisfy the Threshold condition will done.
- Forward the result for further XOR Operation
- If Bitwise_XOR_Result>T then put the combination in final ICEBERG RESULT List. Also generate new vector.
- Else discard the combination from the list.
- Repeat this operation till List will empty.

Same logic is applicable whenever new vector is generated after XOR operation. In this way to implement all type of aggregate operations like COUNT,SUM,MIN and MAX we have used above logic only as per aggregate functions the sequence of the operation to be change .

C. Performance Evaluation

The objective of this research is to improve the efficiency of IBQ using LAMS. This algorithm is work on the BI created on the attributes of the query. In this experiment first up all we are generating BI and then applying LAMS.The efficiency of this algorithm is measured in terms of number of iterations and time required to execute the query. We have conducted the experiment on different size databases and checked the performance of SUM and COUNT aggregate function. We have compare the results with previous techniques such as bitmap indexing strategy(BIS) and Dynamic pruning strategy (DPS). We found significant improvement in case of LAM Strategy.

Performance of all above strategies is measured in terms of time. The dataset from IBM Watson community group is used. We apply the algorithm on different tuple size dataset. Similarly we have written different Iceberg queries and performance is measured.

Query 1:
Select Month, Category, Count(*) from Monthly_Exp_25 group by Month, Category having Count(*)>=4;
Query 2:
Select Month, Category, SUM(Amount) from Monthly_Exp_25 group by Month, Category having SUM(Amount)>1000;

We noticed significant improvement in performance as we go on increasing the size of database as well as threshold value(T).Following Figure 2,Figure 3,Figure 4,Figure 5 shows the comparison of look ahead matching(LAM) strategy with vector alignment (VA) and Dynamic Pruning(DP) for IBQ evaluation. In Figure.3 and Figure 4 we noticed that even we increase database size as well as threshold then also number of iterations to execute query get reduced and remain constant for increase T. In this way we have perform the experiment on different Data sets, different aggregate functions and different threshold values.

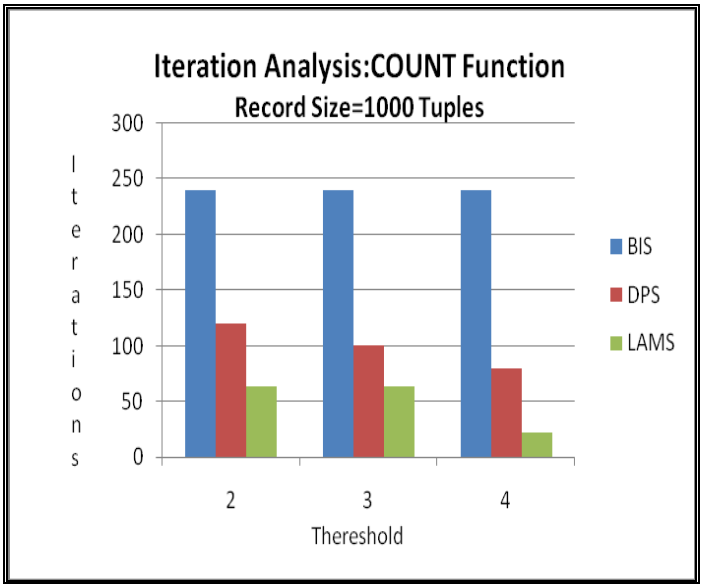

Figure 2: COUNT Function on 1000 Tuple Dataset

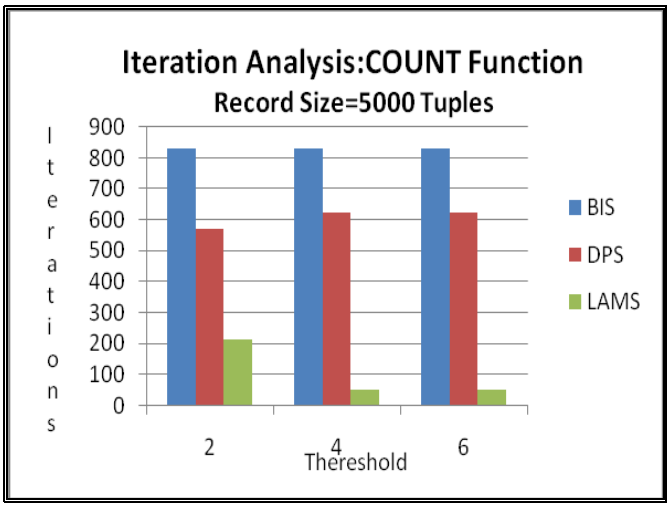

Figure 3: COUNT Function on 5000 Tuple Dataset

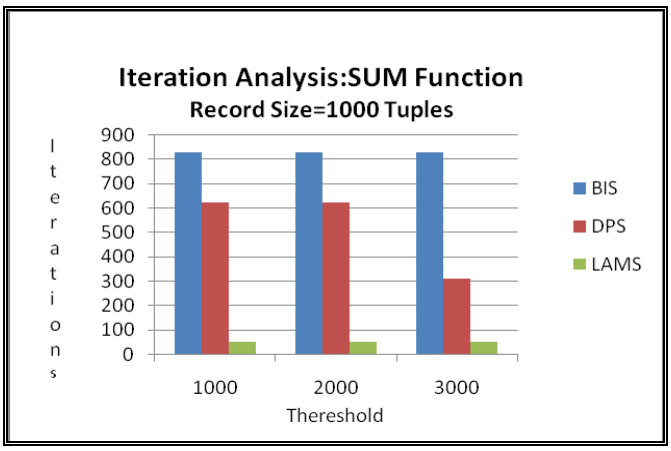

Figure 4: SUM Function on 1000 Tuple Dataset

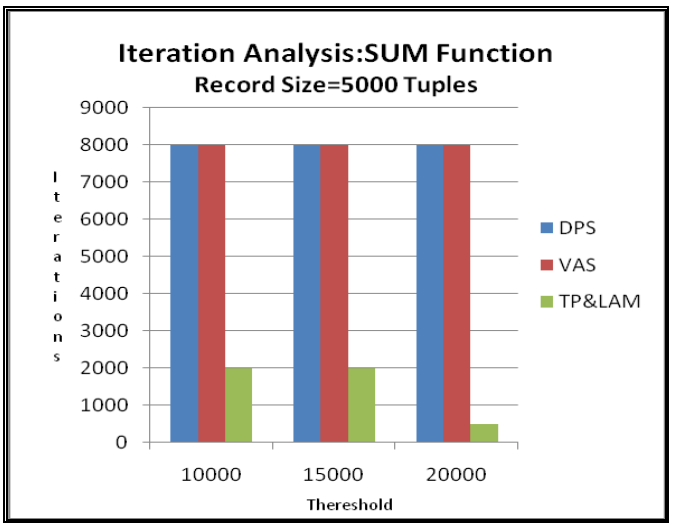

Figure 5: SUM Function on 5000 Tuple Dataset

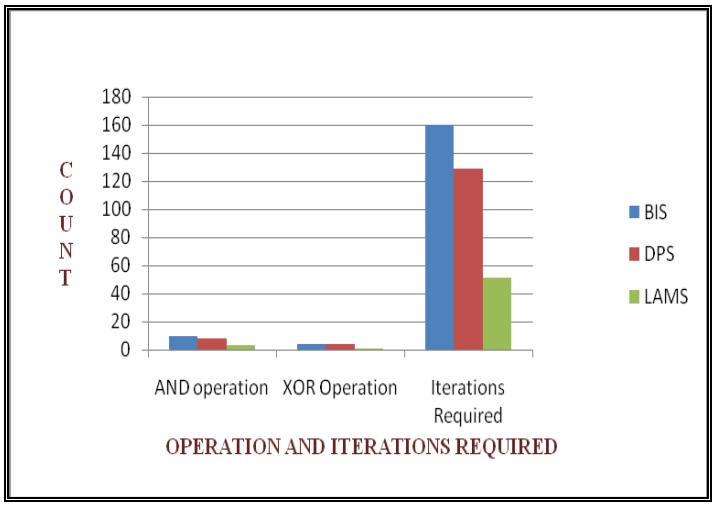

Figure 6: Analysis of query1

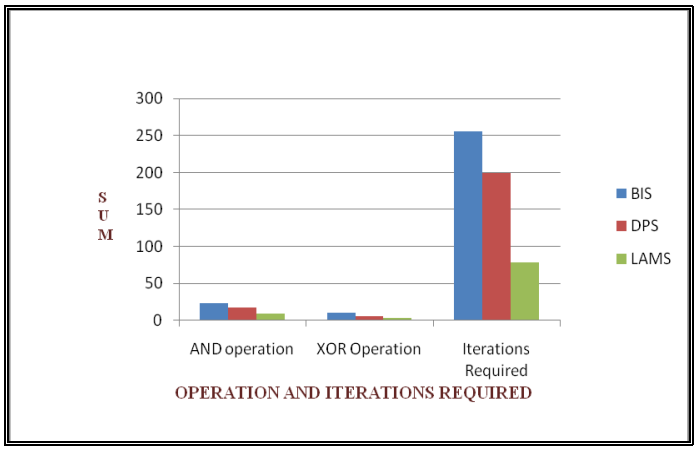

Figure 7: Analysis of query2

Iteration analysis is represented in Figure 6 and Figure 7.Here we observed that as per the requirement of aggregate function the number of iterations get varied from function to function. The performance of LAM strategy in terms of iteration, AND and XOR operation required is improved. As the iterations required get reduced it directly improves the performance in terms of time.We have also tested query 1 and query 2 on different size of data set.In that case also we found the performance of LAM strategy superior to all traditional algorithms.

IV. CONCLUSION

To execute complex OLAP queries on DW require much processing cost. Normally, intention of DW query is analyzing some parameters against another parameters from same database. Almost for all analysis task aggregation is the main function. Processing and executing queries which contain aggregate function in traditional way is time consuming process which require more I/O access . In this

paper we have proposed LAMS which help to execute such a complex queries efficiently. Our experimental results help us to prove the superiority of our research. The result of this research will help to execute quires with aggregate function as well as IBQ which improve the performance of OLAP queries on DW.This research concentrate only on COUNT and SUM aggregate function.In future by extending this concept we will develop framework for other aggregate functions such as MIN,MAX and AVERAGE.The focus of this research is only structured database but in future we can apply the same logic for query processing on unstructured data.

REFERENCES

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

[2] Kazi, Z., B. Radulovic, D. Radovanovic, and Lj Kazi. "MOLAP data warehouse of a software products servicing Call center." In MIPRO. 2010 Proceedings of the 33rd Inter national Convention, pp. 1283-1287. IEEE, 2010

[3] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman," Computing iceberg queries efficiently" VLDB Conf., pages 299-310, 1998.

[4] Bin He, Hui-I Hsiao, Ziyang Liu, Yu Huang and Yi Chen, "Efficient Iceberg Query Evaluation Using Compressed Bitmap Index", IEEE Transactions On Knowledge and Data Engineering, vol 24, issue 9, sept 2011, pp.1570-1589

[5] Parth Nagarkar,"Compressed Hierarchical Bitmaps for Efficiently Processing Different Query Workloads",IEEE International conference on Cloud Engineering ,DOI 10.1109/IC2E.2015.99

[6] C.V.Guru Rao, V. Shankar,"Efficient Iceberg Query Evaluation Using Compressed Bitmap Index by Deferring Bitwise- XOR Operations "978-1-4673-4529-3/12/$31.00c 2012 IEEE

[7] C.V.Guru Rao, V. Shankar, "Computing Iceberg Queries Efficiently Using Bitmap Index Positions" DOI: 10.1190/ICHCI-IEEE.2013.6887811 Publication Year: 2013 ,Page(s): 1 – 6

[8] Vuppu.Shankar, Dr.C.V.Guru Rao," Cache Based Evaluation of Iceberg Queries", IEEE International conference on Computer and CommunicationsTechnologies(ICCCT),2014,DOI: 10.1109/ICCCT2.2014.7066694 ,Publication Year: 2014

[9] Rao, V.C.S. , Sammulal, P.," Efficient iceberg query evaluation using set representation",India Conference (INDICON), 2014 Annual IEEE DOI: 10.1109/INDICON.2014.7030537 Publication Year: 2014 , Page(s): 1 – 5

[10] K.-Y. Whang, B.T.V. Zanden, and H.M. Taylor, "A Linear-Time Probabilistic Counting Algorithm for Database Applications," ACM Trans. Database Systems, vol. 15, no. 2, pp. 208-229, 1990

[11] 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

[12] K.P. Leela, P.M. Tolani, and J.R. Haritsa, "On Incorporating Iceberg Queries in Query Processors" Proc. Int'l Conf. Database Systems for Advances Applications (DASFAA), pp.431-442, 2004

[13] Ying Mei, Kaifan Ji*, Feng Wang," A Survey on Bitmap Index Technologies for Large-scale Data Retrieval" 978-1-4799-2808-8/13 $26.00 © 2013

[14] F. Delie`ge and T.B. Pedersen, "Position List Word Aligned Hybrid: Optimizing Space and Performance for Compressed Bitmaps," Proc. Int'l Conf. Extending Database Technology (EDBT), pp. 228-239, 2010

[15] A. Ferro, R. Giugno, P.L. Puglisi, and A. Pulvirenti, "BitCube: A Bottom-Up Cubing Engineering,"Proc. Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK), pp. 189-203, 2009

[16] R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules Between Sets of Items in Large Databases. ACM SIGMOD Conf. Management of Data, pages 207-216, 1993

[17] W., Perrizo, Peano Count Tree Technology, Technical Report NDSU-CSOR-TR-01-1, 2001

Authors Profile

Dr. S.N.Zaware pursed Bachelor of Computer Engineering from University of Pune in 1999 , Master in Computer Science and Engineering from Swami Ramanand Theerth Marathwada University in year 2005 and Ph.D. Computer Science and Engineering from St.Peters Institute of Higher Education and Research Chennai in 2018. She is a Life member of CSI,ISTE and IAENG. She has published more than 20 research papers in reputed international journals including SCOPUS Indexed Journal and conferences including IEEE and SPRINGER it's also available online. Her main research work focuses on Query optimization ,data ming ,machine learning and Big Data Analytics. She has 19 years of teaching experience and 2 years of Research Experience.

