International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 17 (2017) pp. 6542-6552
© Research India Publications. http://www.ripublication.com

Priority and Probability Based Model to Evaluate Aggregate Function Used in Iceberg Query

Kale Sarika Prakash
Research Scholar, Department of Computer Science and Engineering,
St.Peter's Institute of Higher Education and research, St.Peter's University, Avadi, Chennai, India.
Orcid Id: 0000-0001-5033-9609

Dr. P.M.Joe Prathap
Research Supervisor, Department of Computer Science and Engineering,
St.Peter's Institute of Higher Education and research, St.Peter's University, Avadi, Chennai, India.
Orcid Id: 0000-0002-4708-8308

Abstract

Data warehouse is collection of huge data set. Data from data warehouse is analyzed using iceberg query and OLAP function .The objective of this research is to improve the performance of iceberg query in terms of time and I/O access. Iceberg query is complex as it contain aggregate function followed by HAVING and GROUP BY clause. So, efficient execution of aggregate function is very important for better performance of iceberg query. None of the previous research provides the model for all aggregate functions and all of them are not efficient because they faces the problem of empty bitwise AND, OR and XOR operation . Proposed research uses priority and probability based model on bitmap index to evaluate the aggregate function used in iceberg query. This help to reduce the time and I/O access cost of evaluating iceberg query. Experimental result shows improvement in the performance of iceberg query using priority and probability based model. The performance is increase [60-70] % for COUNT and SUM function and [10-20] % for MIN and MAX function. This research provide framework for aggregate functions like MIN, MAX, COUNT and SUM. In this way this research tries to improve the performance of iceberg query which is essential in large data analysis.

Keywords: Aggregate functions: (COUNT, SUM, MIN, MAX), Bitwise operations: (AND, OR,XOR), Bitmap Index (BI), Iceberg Query (IBQ),priority and probability based model(PPBM)

INTRODUCTION

Basically, data warehouse (DW) users are the business analyst, manager of different department, decision maker, and knowledge worker of organization. They analyze the data from data warehouse and forecast some issues related to their business. Based on these predictions they decide some business policies. Once policy is frame they implement it and observe its progress as per their forecasting. This observation phase require more time to validate its outcome. To perform all these activities they have to work on huge, historical and subject oriented data set which is known as data warehouse. The information which they collect from DW is small information from huge dataset. To extract such a type of data the query to be executed is performing aggregation function on some specified condition with some threshold value. This type of query is called as IBQ.

IBQ were first studied by scientist named Min Fang et.al [1].According to him, an iceberg query has lot of application in DW, data mining and information retrieval systems.IBQ is defined as the type of query which perform aggregation function on set of attribute followed by having clause on some condition or on threshold value. Because of having clause the aggregate function which does not satisfy threshold condition will get eliminate from the result. In this way the output of iceberg query is small set of data from huge input data set. Because of this type of nature it is called as Iceberg.

Syntax of an Iceberg queries on a relational table R1 (Col1, Col2,,… ,Col n) is stated below:

SELECT Col 1, Col 2, …, Col n, AGG(*)

FROM R1,

GROUP BY Col 1, Col 2, …, Col n,

HAVING AGG (*) > = T

Where Col 1, Col 2, …, Col n represents a subset of attributes in R1 which are aggregate attributes. AGG represents an aggregation function such as MIN, MAX, COUNT, SUM and AVG. The predicates used in HAVING clause are greater than equal to(>=), less than or equal to (<=) or is equal to (= =) .

Due to threshold condition IBQ return very small distinct data set as output which looks like the tip of an iceberg. As output of iceberg query is very small compare to input, it can answer quickly even on huge data set. Our research makes use of this feature of IBQ. However, current database systems do not fully take advantage of this feature of IBQ. The relational database systems like Oracle, MYSQL, SQL server,DB2, and column-oriented databases[2],[3],[4]all are using general aggregation algorithms to execute IBQ. They did not consider it as special query so the time required to execute such queries on these databases are more. They answer IBQ by first aggregating all tuples and then evaluating the HAVING clause to generate final iceberg result.

The main objective of this research is efficient execution of aggregate function of IBQ. Performance of IBQ is completely depends upon the aggregate function used in query and data size on which query is going to work [5] and [6]. To achieve this objective this research proposes novel IBQ evaluation strategy which works on priority and probability concept. Based on the query the attribute bitmap vector is evaluated in advance using priority based strategy. After this the vectors which are having highest probability of to be a part of final answer are evaluated. Then comparison of the result with next highest vector is done so the chances of pruning the vector in advance will be more. In this way by performing only required operations it reduces I/O access as well as time required to execute IBQ.This process avoids fruitless bitwise AND, OR and XOR operations problems occur in previous research [7],[8],[9] and [10]. None of the research [7],[8],[9] and [10] work on other aggregate functions like MIN,MAX and AVERAGE. Our research provides a frame work for aggregate functions like SUM, COUNT, MIN and MAX which work on structured database. Our experimental result prove that performance of our strategy is better than previous algorithms. In future by extending this concept to be work on unstructured database it will help for big data analysis also [11].

This paper is organized into following sections. Section II describes review of iceberg query processing strategies and bitmap indexing technique. Section III focuses on priority and probability based model to evaluate aggregate function used in IBQ its workflow diagram and pseudo code. Experimental results with graphical representation are described in section IV and section V concludes the paper.

REVIEW OF ICEBERG QUERY PROCESSING STRATEGIES AND BITMAP INDEXING TECHNIQUE

In this section we are highlighting the IBQ execution strategies developed by different researchers, drawback of these methods and application of bitmap index to execute IBQ. The IBQ processing is first described by Min Fang [1] in 1998.In this research author proposed technique based on sampling and probabilistic count. Based on threshold value the probable sample from data set is collected and processed as per aggregate function. But it requires complete table scan repeatedly due to this query processing cost in terms of time is more in this case. After that IBQ processing is proposed by [12] in 2000. This research concentrates only on evaluation of AVERAGE function used in IBQ by using partitioning technique. Main concept behind this technique is to partition database logically to find candidate set of tuples which are satisfying threshold condition of IBQ. They work in two phases first partition the relation and selecting candidate and second phase which computes average value of candidate set. The performance of this algorithm depends upon data order of tuple and memory size. If the table is in sorting order then the performance of above strategies are excellent without regards to memory size. Existing optimization techniques for processing IBQ suggested by [1] and [12] can be categorized as the tuple-scan-based method which requires minimum one table scan to read data from disk. They concentrate on reducing the number of table scan required when dataset size is large. They did not make use of IBQ property for efficient query processing. Due to this a tuple-scan-based method takes a long time to answer query, when the dataset is very large.

Framework for IBQ processing has proposed by [13] in 2004. In this paper researchers developed different IBQ processing technique and suggest algorithms that can be implemented in query optimizer to choose more relevant algorithm for IBQ evaluation.

Evaluating query by finding sub query is proposed in [14].The basic intention of this research is to optimize the query evaluation process. But the limitation of this approach is it is not suitable for the application where data retrieval rate is more.

Above mentioned approaches suggested by [1],[12],[13] and [14] are comes under the group of tuple scan based method. None of the above research considered properties of IBQ for its evaluation. These algorithms focus on reducing number of tuple scan but none of them makes use of IBQ property.

Bitmap index is usually a better choice for querying the massive and multidimensional scientific datasets [24,25,26]. It has significantly increases data accessing time and reduced the query response time on both high and low cardinality values with a number of techniques [7][22]. 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 [8]. Use of bitmap index to execute iceberg query is important because bitmap avoid massive disk access on complete tuple. It access bitmap index of attributes of GROUP BY clause only. It operates on bits rather on actual tuple values. Bitwise operations are very fast to execute as they directly supported by hardware [9][21][23].Bitmap also leverage the antimonotone property of IBQ very easily. This helps for index pruning in IBQ Evaluation [10,11].

Ferro et al. [12] designed a two-level bitmap index which can be leveraged for processing IBQ. By considering applicability of bitmap indexing [12] in 2009 make use of bitmap indexing for IBQ Evaluation. This is the first research which makes use of bitmap indexing for IBQ evaluation. But this strategy suffers from empty bit wise AND, OR and XOR operation problem. This problem is minimized by [13] using dynamic pruning and vector alignment approaches. Both of these approaches make use of the antimonotone property of IBQ.

Research [14] try to handle empty XOR operation problem but did not able to solve fruitless bitwise AND operation problem. Similarly research [15] and [16] also tries to minimize fruitless AND operations. They also not able to minimize XOR operation. All above [13], [14], [15] and [16] uses priority queue concept but all of them faces futile queue pushing problem. The queue maintenance cost is involved in each pass. All faces fruitless bitwise AND, OR and XOR operation problem.

To handle all above problems we are proposing novel IBQ evaluation strategy which works on priority and probability concept. Based on the query the attribute bitmap vector is evaluated in advance using priority based strategy. This strategy first analyse which operation to be perform first 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. In this way by performing only required operations it reduces I/O access as well as time required to execute IBQ. Our experimental result proves that performance of our strategy is better than previous algorithms.

Priority and probability based model for IBQ evaluation

This section highlight the methodology used in our research to improve performance of IBQ. We are using priority and probability based model for IBQ evaluation. This is used to interface with any DW or large database system. This model is working in three phases. Phase I: Bitmap vector generation which is as shown is Figure 1.

This phase concentrate on selection of only required attribute from huge data set as per the content of IBQ. This is very important task as in future it helps to reduces I/O overheads. It select only required attribute from huge input data set and for IBQ processing only these parameters are considered throughout evaluation of IBQ. Other important feature of this module is it create bitmap vector of selected input which is in the form of 0's and 1's which help for fast query processing.

Next phase is Phase II : Priority based model which is as shown in Figure 2.The input for phase II is bitmap vector generated by phase I. Phase II rearrange all the vectors as per their priority. Priority is assigned based on the position of 1's bit appear in vector. Once the priority is assign each vector is located in priority queue. Then it perform bitwise AND operation between highest priority vector. If result of AND operation satisfy threshold condition then it will be included in final IBQ result else it is discarded from the list of input vector. Once it is included in final IBQ, bitwise XOR is performed to generate new vector. Look ahead matching approach is used to find the appropriate place for newly generated vector. According to that the new sequence of priority queue gets formed. Here we are using probability based approach which help us to fit the newly generated vector in appropriate queue which is a Phase III of our module.

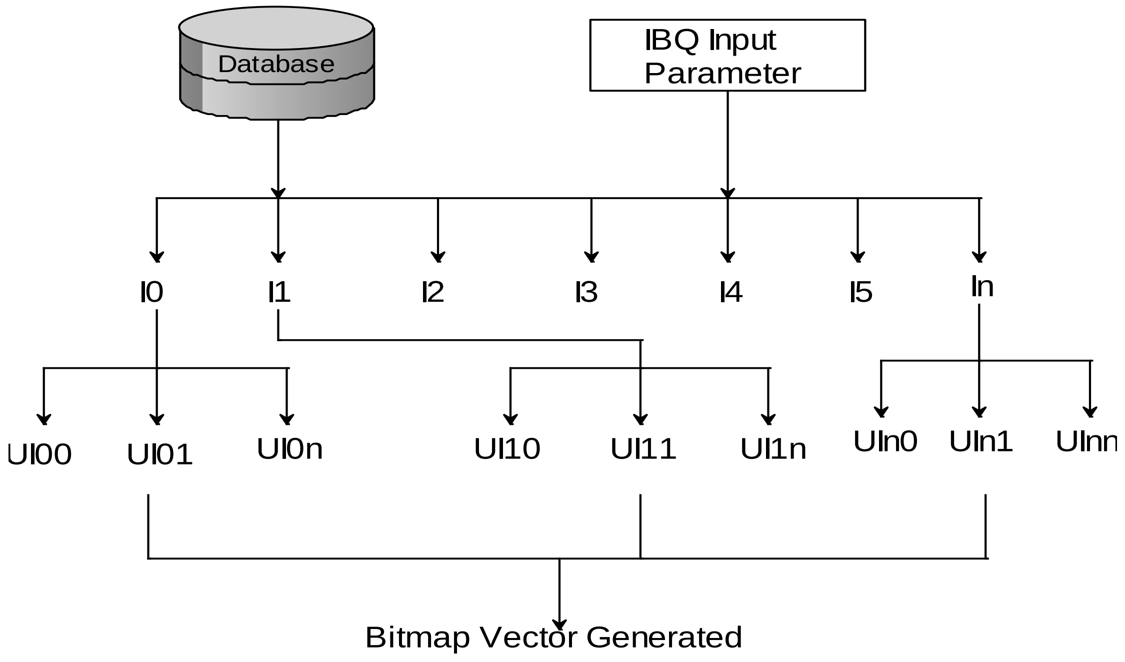

Figure 1: Phase I: Bitmap vector generation

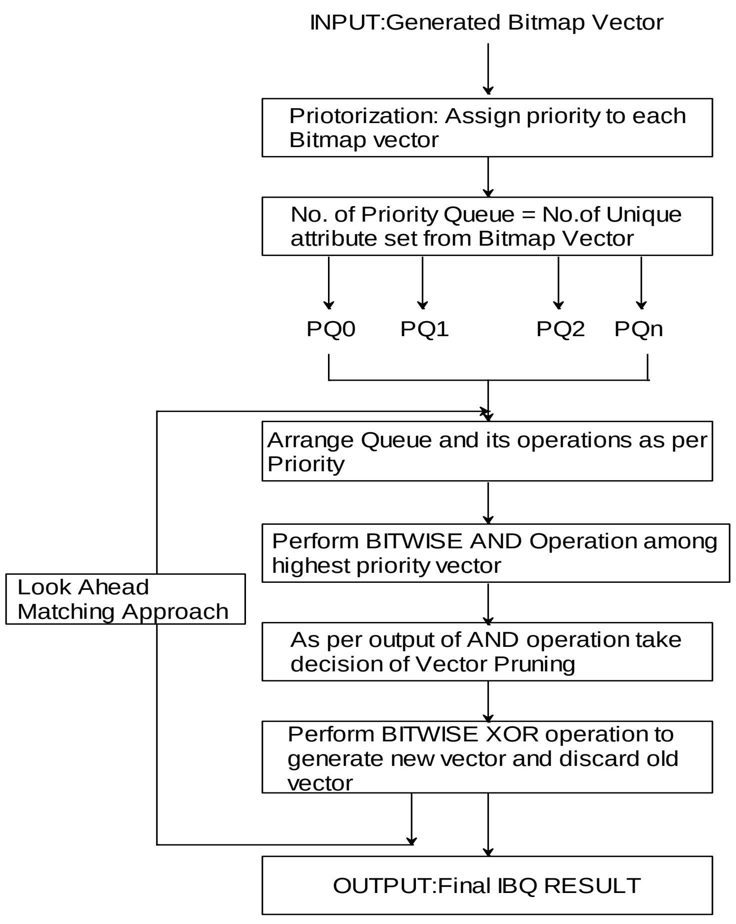
INPUT:Generated Bitmap Vector
Priotorization: Assign priority to each Bitmap vector
No. of Priority Queue = No.of Unique attribute set from Bitmap Vector
PQ0
PQ1
PQ2
PQn
Arrange Queue and its operations as per Priority
Perform BITWISE AND Operation among highest priority vector
Look Ahead Matching Approach
As per output of AND operation take decision of Vector Pruning
Perform BITWISE XOR operation to generate new vector and discard old vector
OUTPUT:Final IBQ RESULT

Figure 2: Phase II: Priority based Model

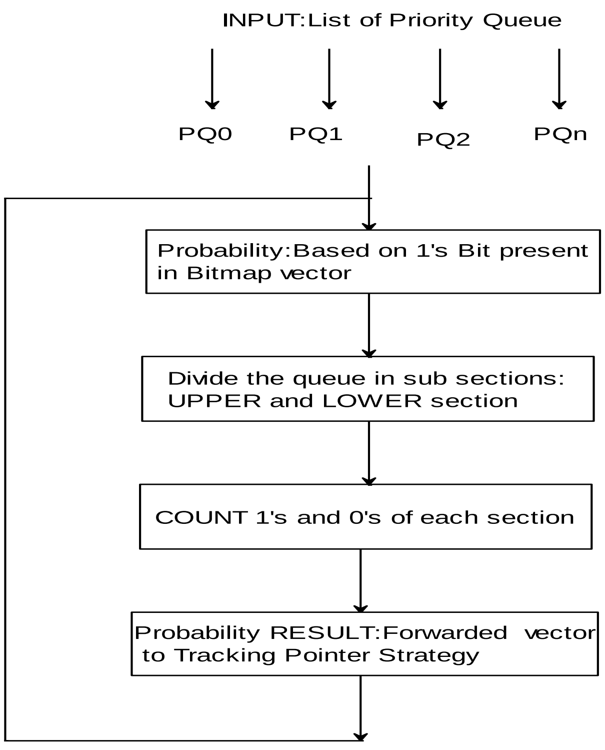
INPUT:List of Priority Queue
PQ0
PQ1
PQ2
PQn
Probability:Based on 1's Bit present in Bitmap vector
Divide the queue in sub sections: UPPER and LOWER section
COUNT 1's and 0's of each section
Probability RESULT:Forwarded vector to Tracking Pointer Strategy

Figure 3: Phase III: Probability based Model

Phase III: Probability based model is as shown in Figure 3. This module is work along with look ahead matching approach. These two concepts work together to find the probability of vector to be part of final result. To find probability it divide the vector into sub sections like UPPER and LOWER part. It finds the probability of LOWER and UPPER vector independently and then matches it with other priority vector. The matching where the results are close to threshold value is treated as most probable vector. Here this phase uses tracking pointer strategy to track the vector continuously till the declaration of its probability. In this way our proposed priority and probability based model work together to evaluate IBQ efficiently.

Implementation detail of priority and probability based model for IBQ evaluation

This section highlights the functionality used in priority and probability based model used for IBQ Processing with some additional concept like tracking pointer and look ahead matching strategy. Input to algorithm is IBQ (attribute A, attribute B, threshold T) and output is IBQ RESULT which is set of combination of vector which satisfy threshold condition.

Initially as shown in Figure1 phase I generates bitmap vector on attribute A and attribute B. On this matrix form of bitmap vector following algorithm will process.

1. First clear all queue to be used in processing. PriorityQueueA.clear, PriorityQueueB.clear upto final queue required.
2. For each vector x of attribute A do
3. if x.count_of_1 >= T then
4. x.next==FirstOneBitPosition(x,0)
5. PriorityQueueA.Push(x)
6. Similarly PUSH the highest priority vector of attribute B in PriorityQueueB
7. x,y = NextAlignedVector(PriorityQueueA.clear, PriorityQueueB,T)
8. Here our strategy initiates first bitwise AND operation between highest priority vector from both the queue.
   i. If both the queue has first 1 bit position at first location then algorithm will forward it for AND operation.
   ii. Tracking pointer strategy will get initiated to track next highest priority vector from both attributes.
   iii. If PriorityQueueA.first.bit=0 then shift the tracking pointer to right and go on checking till first one bit found.
   iv. Above procedure repeated for PriorityQueueB if its first bit is 0.
   v. When we found 1 bit in any queue we will have to set the value of count== current pointer location.
   vi. The pointer of the vector directly point to count and whatever remaining bits from both the vectors are returned for further processing i.e. for AND operation.
9. While x!=NULL &y!=NULL do
10. PriorityQueuea.Pop
11. PriorityQueueA.Pop
12. Result=BitwiseAND(x,y)
13. If(Result.count>=T) then
14. Add Iceberg Result(x.value,y.value,result.count)
15. x.count=x.count-result.count
16. y.count=y.count-result.count
17. If x.count>=T then
18. A.next1=FirstOneBitPosition(x,x.next+1)
19. If x.next1!=NULL then
20. PriorityQueueA.Push(x)
21. Repeat step 10-20 for vector B
22. PriorityQueueB.Push(y)
23. x,y=NextAlignedVector(PriorityQueueA, PriorityQueueB,T)
24. Return Result
25. Here the look ahead matching concept is used . Once priority of vector is decided we can decide from that which vector to be perform which operation first. This helps us to prune the vector in advance. This step avoid fruitless AND operation by using XOR operation.
26. Similarly to reduce fruitless XOR operation we are using Look ahead matching strategy.
27. Repeat the all above steps till all the vectors are empty.
28. GENERATE ICEBERG RESULT

In this way the complete strategy is work along with tracking pointer and look ahead matching approach. This strategy is implemented using JAVA 7.0 and backend ORACLE 10g.In this way we have developed the framework for aggregate functions like MIN, MAX, COUNT and SUM.

EXPERIMENTAL RESULTS AND DISCUSSION

This section represents the performance analysis of IBQ .For measuring the performance of query we have considered number of AND operation, XOR operation, number of iterations and time required to execute IBQ. We have measured these parameters against different database size, threshold value and different aggregate functions. For experimentation we have used synthetic data set of tuple size

5K, 10K, 20K, 40K and 80K. In this section we are assuming some sample of experimental result for analysis. This section is subdivided into following parts. In all cases we have compared the result of our priority and probability based model (PPBM) with previous strategies like dynamic pruning strategy (DPS) and vector alignment strategy (VAS).

IBQ evaluation based on number of AND and XOR operation

We have applied our priority and probability based algorithm to evaluate query I and query II as shown in Table 1 and Table 2. Query I and query II are the iceberg query. We have done experimentation for above queries using previous strategies like dynamic pruning strategy(DPS),Vector alignment strategy(VAS) as well as our proposed priority and probability based model(PPBM) for IBQ evaluation. We recorded the results and represented it graphically as shown in Figure 4. We counted number of AND operation, XOR operations and iteration required to solve query using our approach. In this case we noted that with traditional algorithm more than 80% operations are useless which directly affect on the performance of IBQ. In case of PPBM the requirement of performing AND and XOR operation is reduced to 40% compare to old algorithms. Because of this time required to execute query get reduced in PPBM.

Query I :Select X, Y, count (*) from R Group By X,Y having count(*)>3 ;

Table 1 : Relation R and its bitmap index of X,Y

X	Y	Z	X1	X2	X3	Y1	Y2	Y3
X1	Y2	500	1	0	0	0	1	0
X3	Y1	600	0	0	1	1	0	0
X2	Y2	100	0	1	0	0	1	0
X3	Y1	500	0	0	1	1	0	0
X2	Y2	100	0	1	0	0	1	0
X3	Y1	600	0	0	1	1	0	0
X1	Y2	500	1	0	0	0	1	0
X2	Y2	600	0	1	0	0	1	0
X1	Y1	100	1	0	0	1	0	0
X2	Y2	500	0	1	0	0	1	0
X3	Y1	600	0	0	1	1	0	0
X2	Y3	100	0	1	0	0	0	1
X1	Y1	600	1	0	0	1	0	0
X3	Y3	100	0	0	1	0	0	1
X3	Y2	100	0	0	1	0	1	0

Query II :Select A, B ,Count (*) from R Group By A,B having count(*)>2;

Table 2 : Relation R and its bitmap index of A,B

A	B	C	A1	A2	A3	B1	B2	B3
A2	B2	1.23	0	1	0	0	1	0
A1	B3	2.34	1	0	0	0	0	1
A2	B1	5.56	0	1	0	1	0	0
A2	B2	8.36	0	1	0	0	1	0
A1	B3	3.27	1	0	0	0	0	1
A2	B1	9.45	0	1	0	1	0	0
A2	B2	6.23	0	1	0	0	1	0
A2	B1	1.98	0	1	0	1	0	0
A1	B3	8.23	1	0	0	0	0	1
A2	B2	0.11	0	1	0	0	1	0
A3	B1	3.44	0	0	1	1	0	0
A3	B1	2.08	0	0	1	1	0	0

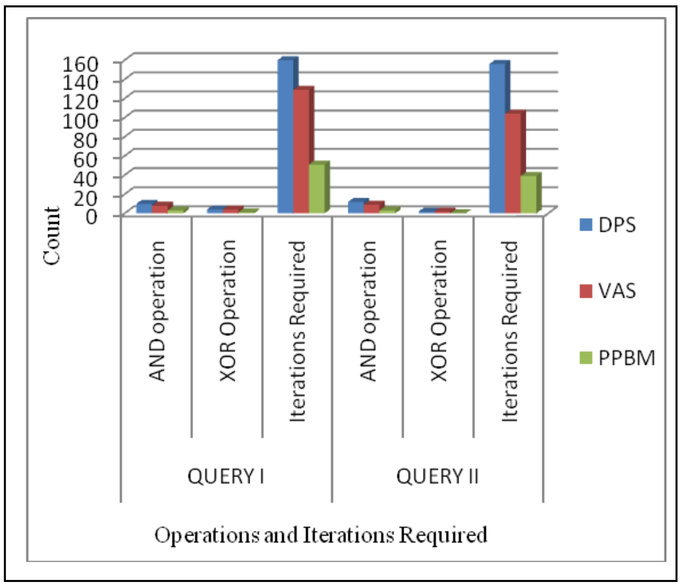

Figure 4: Result of Query I on table I and Query II on table II

IBQ evaluation based on threshold value, iterations required and database size

To evaluate the performance of our strategy against threshold size and data set size we have executed different IBQ with different aggregate functions .In this subsection we have analyzed the performance of IBQ using COUNT and SUM function which is as shown in Figure 5 to Figure 9. We perform the experiment on synthetic data set of tuple size 5K, 10K, 20K and 40 K. To observe the effect of query performance we have assumed the input data as double of previous input dataset like 5K, 10K, 20K and 40 K.

Figure 5 to Figure 9 we observe that performance in terms of iteration required is better through our and probability based model(PPBM) compare to old strategies like dynamic pruning strategy(DPS) and vector alignment strategy(VAS). In case PPBM if we go on increasing data size the iterations required get reduced. The basic objective of our research is to extract small set of data from huge database. As we are extracting small data so time required for extracting such a small data must be less. We observe that number of iterations required get reduced 40 to 50 % even though database size increases to double. From this analysis we have prove that we have achieved our objective.

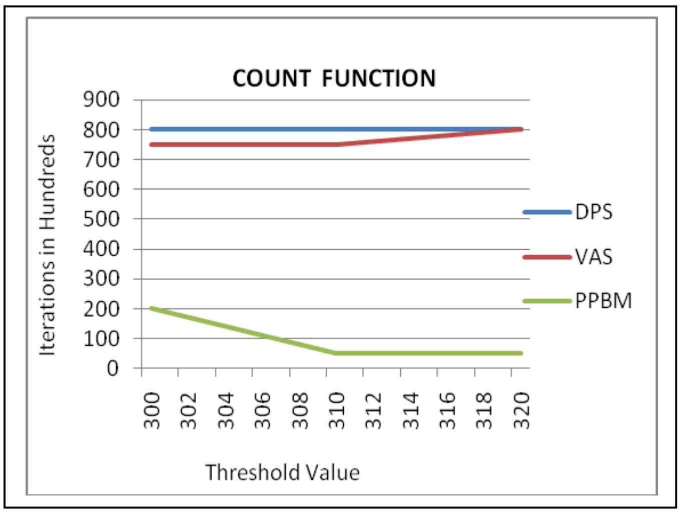

Figure 5. Iteration analysis against threshold value for data set size 5K

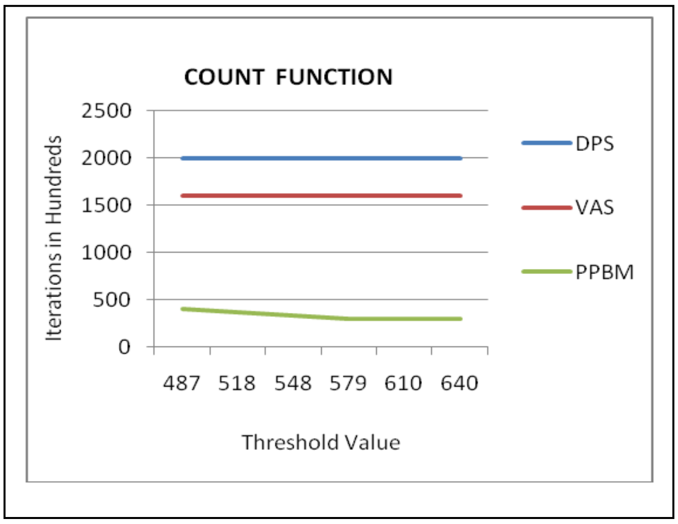

Figure 6. Iteration analysis against threshold value for data set size 10K

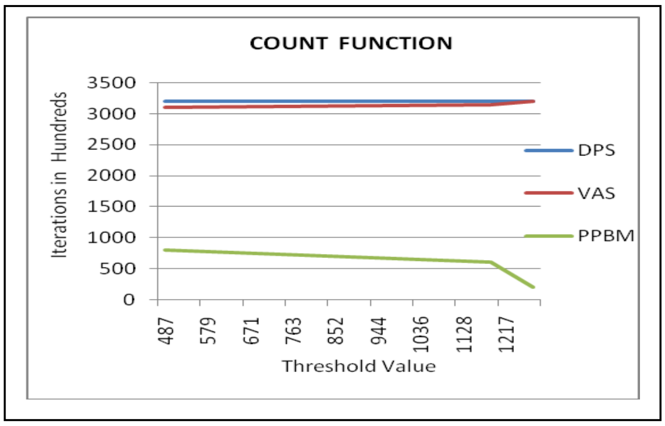

Figure 7. Iteration analysis against threshold value for data set size 20K

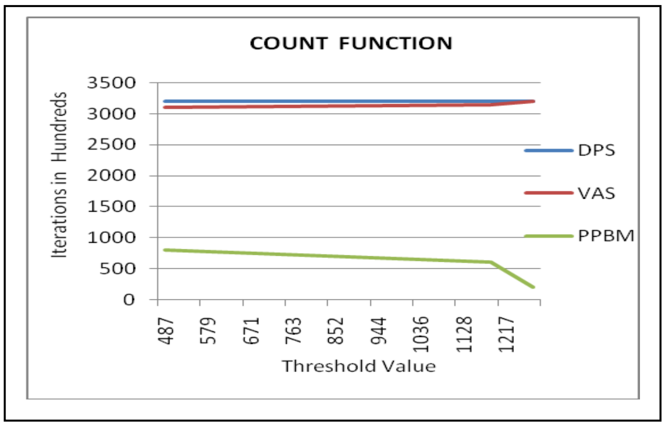

Figure 8. Iteration analysis against threshold value for data set size 5K

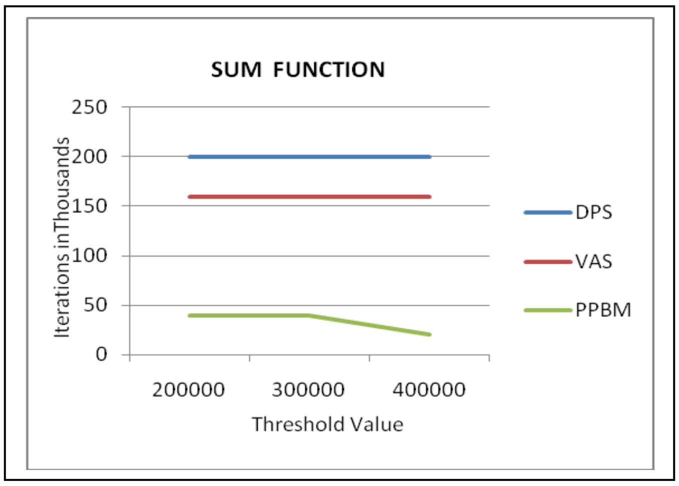

Figure 9. Iteration analysis against threshold value for data set size 10K

IBQ evaluation based on threshold value, database size and time required

To evaluate performance of IBQ in this section we are considering time required to execute query against data set size as well as threshold value. Here we observe that even though we go on increasing the size of dataset then also performance of our strategy is better as we have used BI which reduces data access time. In case of change in threshold also performance is better this is because of TP and LAM strategy. These strategies identify the fruitless operation in advance and avoid processing of such operation due to this even though threshold value and data size increases then also time required to execute query get reduced. The performance of SUM and COUNT function is shown in Figure 10 and Figure 11. We observe that through our PPBM model time required to evaluate IBQ get reduced 60-70% even though data size increases.

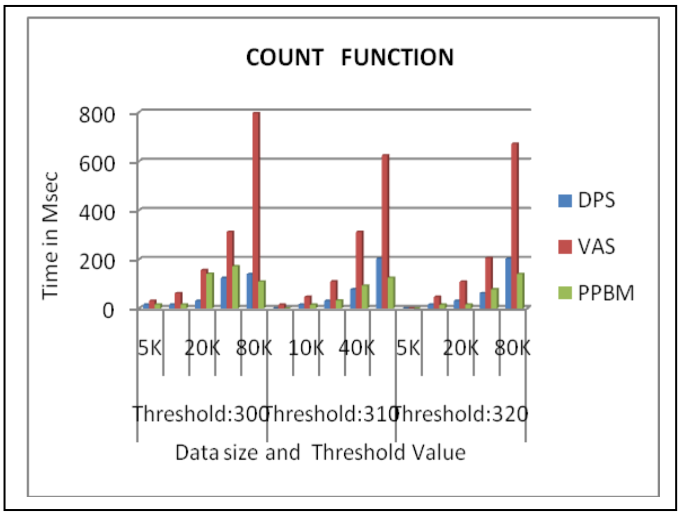

Figure 10. COUNT Function: Time analysis against data size and threshold value

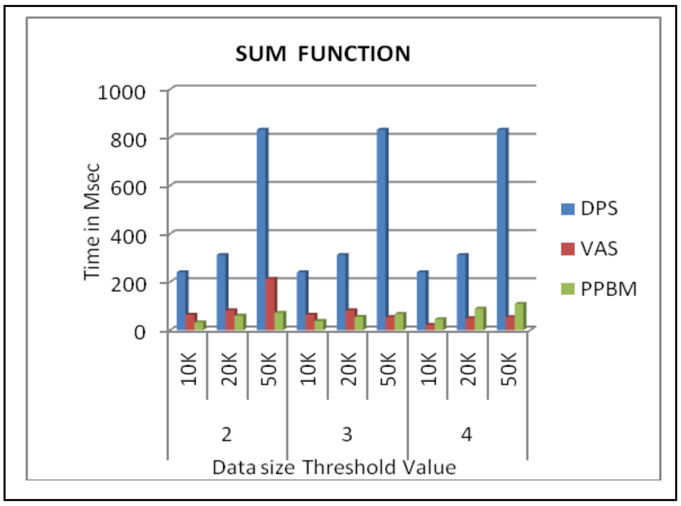

Figure 11. SUM Function: Time analysis against data size and threshold value

IBQ evaluation based on type of aggregate function

In this subsection we are highlighting the important parameter which affect on the performance of IBQ. The aggregate function used in IBQ also has impact on performance of IBQ. Practically, implementation point of view if we consider COUNT, MIN and MAX the evaluation is straightforward but in case of SUM function it is complicated. It requires more operations to evaluate the result. The nature of data present in database is also having close impact on the performance of IBQ. If the data which we want to extract is uniformly distributed among the database then it takes more time to access data. But if there is unequal distribution then it will be fast as per our logic. Our tracking pointer strategy assigns the priority to vector. Priority is assigned as per the probability of that vector. While processing highest priority vector nearby vectors from that group also get verified by the nature of data present in that vector. The vector which is below threshold limit will get directly discarded from list. In this way our PPBM strategy is more efficient if data distribution is unequal. This parameter is beyond the control of our strategy and in such a cases performance gets varied with different aggregate function. This impact which we have observe in as shown in Figure 12 and Figure 13.This figure shows the iteration analysis of MIN and MAX function. We have perform experimentation on all data set like 20K,40K and 80K but here we have shown only for 20K data set.

Here we can see that in case of MIN function performance of PPBM is better compare to DPS and VAS which has impact of our logic that even though threshold increases iterations get reduced. But reduction rate is only 10-20% whereas in case of COUNT and SUM reduction rate is 40-50 %.Important observation related to MAX function is compare to old strategy performance of PPBM is better but for increase in threshold there is no reduction in iterations required.

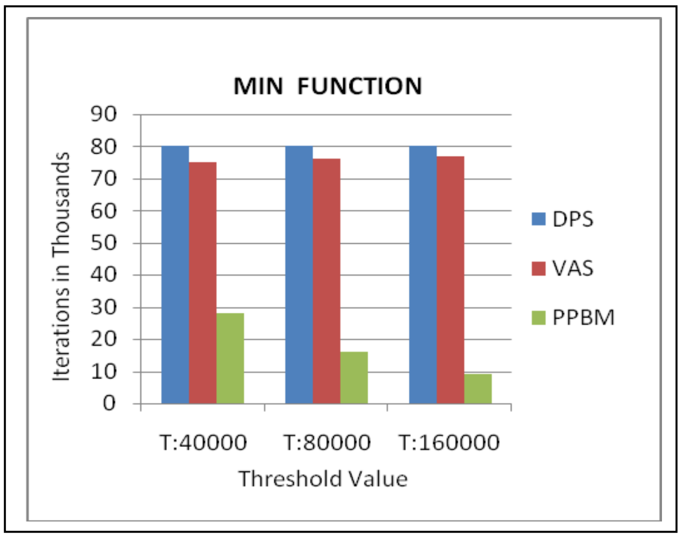

Figure 12. MIN Function

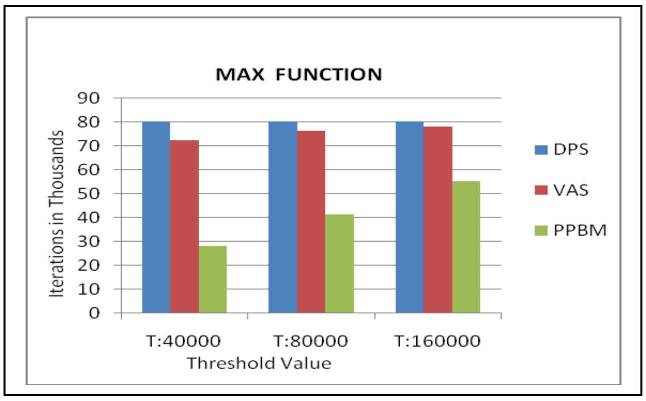

Figure 13. MAX Function

In this way we have done the experimental analysis of IBQ with all aggregate functions on different size synthetic data set. In all cases we found performance of our proposed PPBM strategy is better than all the previous strategies for IBQ evaluation. Thus we prove the superiority of PPBM strategy compare to DPS and VAS.

IBQ evaluation based on number of attributes
This subsection shows the performance of PPBM strategy against the complex input database. Here, complexity of input database is increased by increasing the number of attributes present in the database along with database size. As shown in Figure 14,15,16,17 we can observe that the performance of PPBM strategy is better than previous strategy even though the complexity of input database increases. In this case we observe that there is [70-80] % improvement in the performance of IBQ through our PPBM as number of iterations required to evaluate IBQ is reduced drastically in case of PPBM. These results are due to IBQ property which we have inculcates in our implementation. Similarly we have recorded the performance for all other functions like SUM, MIN, MAX and AVG. In all functions the performance of our strategy is excellent.

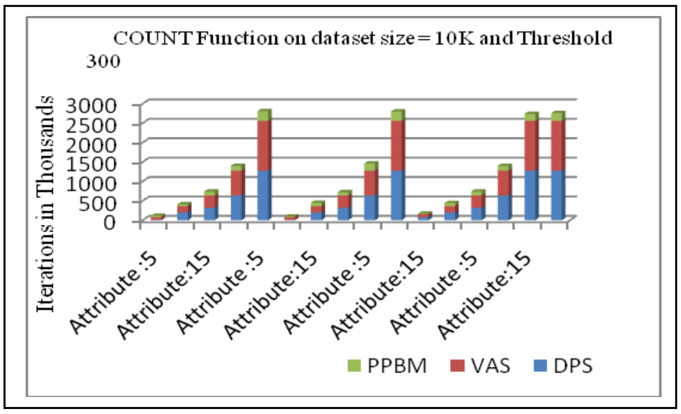

Figure 14. COUNT Function: Iteration analysis against No. of attributes on 10K dataset

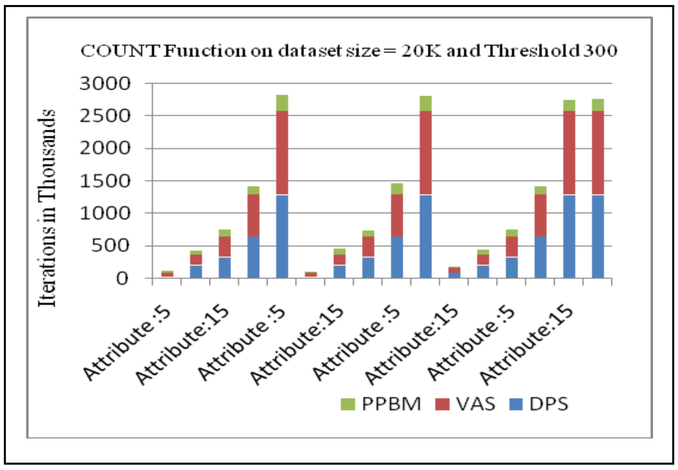

Figure 15. COUNT Function: Iteration analysis against No. of attributes on 20K dataset

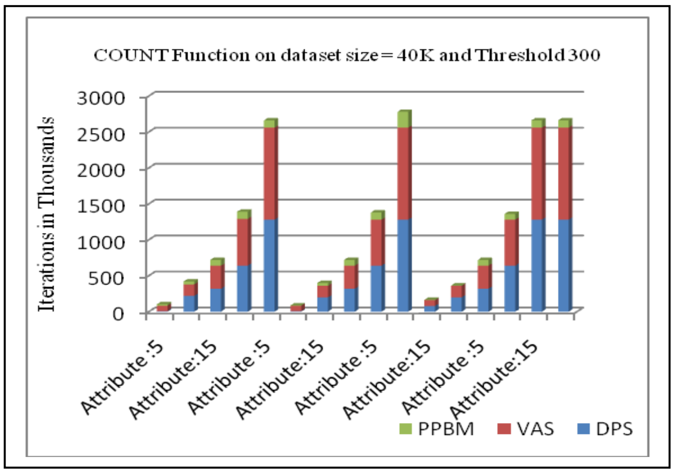

Figure 16. COUNT Function: Iteration analysis against No. of attributes on 40K dataset

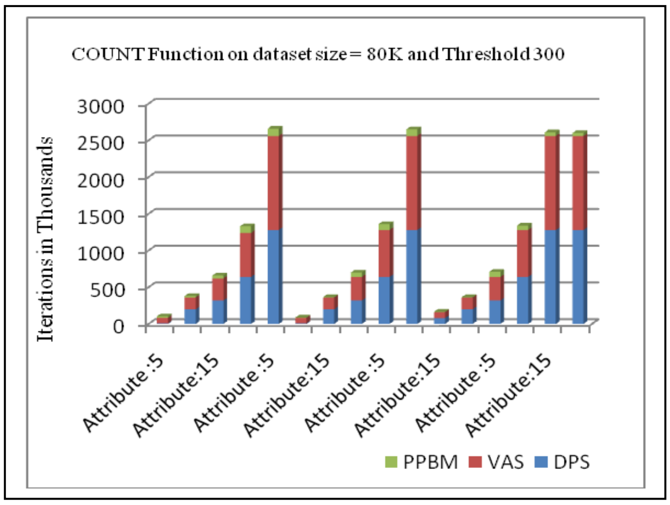

Figure 17. COUNT Function: Iteration analysis against No. of attributes on 80K dataset

CONCLUSION

Basic requirement of Decision support and knowledge discovery systems is to compute aggregate values of interesting attributes by processing a huge amount of data. Proposed PPBM strategy is providing efficient solution to evaluate aggregate function of IBQ and to extract small data from huge database. Our research makes use of bitmap indexing strategy for IBQ processing and for evaluating aggregate function. It uses tracking pointer concept and look ahead matching approach which helps to increase the speed of IBQ. Through our PPBM strategy the performance of IBQ with COUNT and SUM operation is increased 40-50% whereas in case of MIN and MAX it is increased 10-20 %.Thus our experimental results prove the superiority of our strategy by comparing it with previous approaches. The challenge in front of current research is to develop framework for AVERAGE aggregate function. Also this research concentrate only on structured data by extending this concept for unstructured data our strategy will helpful for big data analysis.

REFERENCES

[1] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D. Ullman, "Computing Iceberg Queries Efficiently," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 299-310, 1998 .

http://ilpubs.stanford.edu:8090/423/1/1999-67.pdf

[2] G. Graefe, "Query Evaluation Techniques for Large Databases," ACM Computing Surveys, vol. 25, no. 2, pp. 73-170, 1993 DOI:10.1145/152610.152611

[3] W.P. Yan and P.A. Larson, "Data Reduction through Early Grouping," Proc. Conf. Centre for Advanced Studies on Collaborative Research (CASCON), p. 74, 1994.DOI:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.1959&rep=rep1&type=pdf

[4] P.A. Larson, "Grouping and Duplicate Elimination: Benefits of Early Aggregation," Technical Report MSR-TR-97-36,

DOI:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.4256

[5] Yufei Tao, Cheng Sheng, Chin-Wan Chung and Jong-Ryul Lee," Range aggregation with set selection" , IEEE Transaction on Knowledge and data engineering, Vol.26, No.5,May 2014

[6] Xiaochun Yun, Guangiun Wu,Guangyan Zhang, Kegin Li and Shupeng Wang,"FastRAQ:A Fast Approach to Range-Aggregate Queries in Big Data Environments" IEEE Transaction on Cloud Computing, Vol.3, No.2, April/June 2015.

[7] 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, September 2011, pp.1570-1589

DOI: 10.1109/TKDE.2011.73

[8] 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.DOI: 10.1109/IAdCC.2013.6514418

[9] 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.

DOI: 10.1109/ICHCI-IEEE.2013.6887811

[10] RaoV.C.S, Sammulal,P.,"Efficient iceberg query evalu ation using set representation", Conference (INDICON), 2014

DOI: 10.1109/INDICON.2014.7030537

[11] Gian Antinio Susto,Andrea Schirru,Simone Pampuri and Sean Mcliine ,"Supervised Aggregative Feature Extraction for Big Data Time Series Regression",IEEE Transaction on Industrial Informatics,1551-3203©2015 IEEE, DOI:10.1109/TII.2015.2496231

[12] 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

http://delab.csd.auth.gr/~manolopo/oikonomiko/bae.pdf

[13] 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

http://dsl.cds.iisc.ac.in/publications/report/TR/TR-2002-01.pdf

[14] Sheetal Dhande, G.R. Bamonte," Query optimization in OODBMS: Identifying subquery for query", International Journal of Database Management System, Vol.6,No.2,April 2014,DOI:10.5121/ijdms.2014.6204

[15] Zainab Qays Abdulhadi, Zhang Zuping, Hamed Ibrahim Housien," Bitmap Index as Effective Indexing for Low Cardinality Column in Data Warehouse", International Journal of Computer Applications (0975 – 8887) Volume 68– No.24, April 2013

[16] An Oracle White Paper ,Oracle Database 12c-Built for Data warehouse,Oracle, 2015.

http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-dw-best-practies-11g11-2008-09-132076.pdf

An Oracle White Paper ,Oracle Database 11g for Data Warehousing and Business Intelligence, Oracle ,April 2011.http://www.socoug.org/html/Oracle-Exadata-Whitepaper.pdf

[17] 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 IEEE

DOI 10.1109/ICINIS.2013.88

[18] Manos Athanassoulis, Zheng Yan, Stratos Idreos, "UpBit: Scalable In-Memory Updatable Bitmap Indexing", Sigmod'16,2016.ACM,ISBN-978-1-4503-3531-7/16/06, DOI: http://dx.doi.org/10.1145/2882903.2915964

[19] 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

DOI:10.1145/1739041.1739071

[20] ONeil, E., ONeil, P., Wu, K, "Bitmap index design choices and their performance implications", Database Engineering and application Symposium ,2007, IDEAS, 2007.DOI:10.1109/IDEAS.2007.19

[21] Zhen Chen,Yuhao Wen,Junwei Cao,Wenxun Zheng,Jiahui Chang,Yinjun Wu,Ge Ma,Mourad Hakmaoui and Guodong Peng,"A survey of Bitmap Index Compression algorithm for Big data", IEEE Tsinghua Science and Technology, ISSN 1007-0214, Vol.20, Februray 2015.

[22] Adham Mohsen Saeed"A guide to select indexing technology for relational database management System", International Journal of Advanced Research (2016), Volume 4, Issue 3, 603-61

[23] K. Stockinger, J. Cieslewicz, K. Wu, D. Rotem, and A. Shoshani."Using Bitmap Index for Joint Queries on Structured and Text Data" Annals of Information Systems, 3:123, 2009. DOI: 10.1007/978-0-387-87431-9_95, ISSN 2320-5407

[24] Kesheng Wu, Ekow J. Otoo and Arie Shoshani Lawrence Berkeley National Laboratory Berkeley,"Compressing BitmapIndexes for Faster Search Operations", IEEE Computer Society,2002.

DOI:10.1109/SSDM.2002.1029710

[25] 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

https://pdfs.semanticscholar.org/3932/6f6c20a0990dd31a15bb136e30d42058b75c.pdf