OPEN ACCESS
ISSN: 2249-6645

INTERNATIONAL
JOURNAL
OF MODERN ENGINEERING RESEARCH (IJMER)

# An Algorithm to Evaluate Iceberg Queries for Improving The Query Performance

M.Laxmaiah[1], A.Govardhan[2]
[1]Department of Computer Science and Engineering, CMR Engineering College, Kandlakoya (V) Medchal (M), Hyderabad, Telangana, India-501401, laxmanmettu.cse@gmail.com
[2]School of Information Technology, Jawaharlal Nehru Technological University, Hyderabad, Telangana, India-500085, govardhan_cse@yahoo.co.in

**ABSTRACT:** In recent years, the Iceberg query evaluation is attracted more number of researchers, scientists, and decision makers. The reason behind this is demand of scalability and efficiency. Always researchers try to find the best methods of computation which takes limited computing resources for large databases. The Iceberg Query (IB) analysis showing that the IB queries are consuming more time than association rule formation from the datasets. So researchers are working continuously to solve the problem of the IB query evaluation in the domain of data warehousing, data mining, and information retrieval systems. As the result, many novel ideas and techniques have been generated in IB query evaluation process. In this paper, we proposed a novel algorithm to evaluate iceberg queries in order to improve performance of iceberg queries.

**Keywords:** Data warehouse, Iceberg query, Threshold, Sanpling

## I. INTRODUCTION

The rapid increase of the data warehouse (DW) repositories causes evolution special type of queries .These queries have the feature of the very large input when compared to the output. In recent times IB queries are identified as an important query. These are belongs to many categories of different domain of applications **[1]**. These applications can be found in web mining, data mining [2], and information retrieval systems [12]. Queries later have been extended to data cubes[9,10]. IB queries are executed on very large databases of several petabyte in size. The main important methods of course count algorithm includes:

- For every target array of counters are maintained in memory.
- For large datasets it cannot scale well for user specified threshold value T on the sorted relation R on the disk.

The Sampling and Bucket Counting are the building blocks for the proposed algorithm

- ***Sampling*** **-** This technique [15] samples a small number of records from the relation R, aggregates and retrieves the records that relatively go above the T.
- ***Bucket counting*** - Rather than allocating a counter for each value, a counter is allocated for a group of different values. High frequency is needed to split the values into groups. These building blocks generates "*False Positive Values*" that are considered as candidates for the final result but do not exceed the 'T'. Sampling also causes "*False Negatives* Values that should be in the final result but are not found as candidates. The common goal of all the algorithms [8] is to reduce the number of "*false positives*". They can generally be described using these stages:

- Compute a candidate set *f u*sing sampling.
- Execute bucket counting over *R* with some dependency on ***f***.
- Select the values that exceed the threshold using another scan of *R.*

### 1.1 THE PROPERTIES OF IB QUERY ALGORITHMS

The choice of an algorithm for solving the IB query depends on its properties. The desired properties of the IB query algorithms are:

- ***Number of passes***: The number of passes required over the data must be less so that the evaluation time of the query is also becomes less.
- ***Determinism***: The running time of the algorithm can be deterministic or randomized and it can be deterministic in nature.
- ***Accuracy***: This represents the error of the output of the algorithm. Some algorithms produce exact result

(i.e. zero error) and some algorithms produce approximate results with bounded or unbounded error**.**

- ***Sensitivity to the data order***: Some algorithms are sensitive to the order of the data and that may produce very inaccurate result for some orders.
- ***Memory requirement***: When memory is consumed, small memory requirement is preferred. Small requirements are preferable. Also, we can derive a tight upper bound on this requirement for some algorithms.
- ***Management***: Ease of tuning the hard to manage large parameters of the algorithm.
- ***Disk space***: The extendable disk space required by an algorithm**.**
- ***Data distribution***: Some algorithms are designed for certain distributions and they may fail for other distributions**.**
- ***Parallelization property:*** *Some* algorithms are inherently highly parallelizable. In contrast, other algorithms are hard to parallelize.
- ***Materialization***: Some algorithms must work on materialized relations, which may incur additional disk space and increase the overall cost of the algorithm.

**1.2 DISTRIBUTED DATABASE SYSTEMS**

In recent years the demands for distributed database systems (DDSs) have been increased. The basic reason for this is decreasing of hardware cost and increasing of software implementation cost. Some of the important advantages presented by distributed systems are reliability, availability, easily maintainable, and cheap. Some of these advantages raised other difficulties in distributed systems (DSs).The main difficulty is to process the queries capably; where the data needed to processes the queries are stored at multiple locations. Retrieving data from multiple locations involves transmission of data through networks.

The challenging part in DS is to design an efficient query processing techniques to decrease the overall cost including communication cost. The query optimization difficulty can be categorized in to two main approaches:

- Reduce the transfer cost across the network by decreasing the data transfer rate across it.
- Use parallel processing method in order to reduce query response time.

**1.3 CENTRALIZED DATABASE SYSTEMS**

The storage space and processing time are the major constraints in centralized database **[18]**.Conventional data cubes compute the complete group-by partitions for every combination of possible group attributes. The major problem with the cube operator is the computation and storage of large size data. The major drawback of the cube computation [3, 4 5, and 6] is it has exponential number of dimensions. The more number of the algorithms are proposed by researchers to handle the space complexity problem are performing pre computation of group by operation on subset attributes or by using online aggregation**.** Computer analysts are often responsible for computing more number of attribute values by performing aggregate operations on larger databases. An IB query is a unique type aggregation query which computes aggregate values to a user specified T (Threshold) value. The users can extract highly important aggregate values that often carry more significant information for their business. the general representation of an ib query on a relation s *($d_1$, $d_2$... $d_n$) is* specified below:

**select** $d_i$, $d_j$**…** $d_m$**, agg (*) from r group by** $d_i$, $d_j$ **…** $d_m$ **having agg (*) > = t // agg** represents aggregation functions

Where $d_i$, $d_j$ ... $d_m$ *are* representing a subset of attributes in Relation 'r' and these are treated as aggregate attributes. The greater than or equal to (>=) symbol is used to represent a comparison predicate.

The main focus is on IB query with aggregation function like *count, max, min, avg, sum and rank* contains a property called as an anti-monotone property. The IB queries have in a fascinating an anti-monotone property for all its aggregate functions and predicates. Suppose the count of a main group is below Threshold values 'T' then the count of any super group also can be below Threshold value '*T'.*

With increase in datasets size the efficiency of the IB Query goes down drastically in other words IB Queries fail to scale to large datasets and this has been created necessity to discover new type of efficient query evaluation methods in order to process them easily.

An IB query is used for extracting data from a particular database under some specified conditional parameter. Consider the following query, SELECT A, B CONT (*) FROM R GROUP BY COUNT (*) >=2

The above listed query used to retrieve count of A and B from relation R using some conditional parameters, which specifies the count of A, B greater than 2 should be selected. The above listed types of queries are called IB queries. An IB query reduces the time of fetching data from large databases. Priority is given to IB Queries because of the aggregate function used with the Query. The use of IB queries becomes so frequent in new database management systems, because of their precise data extraction in a limited time.

The remaining paper is organized as follows; section 2 presents the brief over view of the related work. Section 3 presents evaluation of iceberg queries followed by description of random 1 Bit algorithm in section 4. The Section 5 concludes the paper.

## II. RELATED WORK

In the recent past, the compressed BMP index technique is proposed for evaluation of IB queries [12]. The BMP index is built based on bitmap vector which consists of attribute values. The approach is gaining more popularity for column and row oriented databases [13]. In this approach, they are exposed the property of BMP index and invented efficient evaluation strategy of eliminating the scanning and processing of bigger database tables. This approach definitely speeds up IB query evaluation. They finished that compressed Bitmap indexing technique is more capable than existing algorithms of IB Query evaluation.

The IB queries are special type of aggregate queries which computes aggregate functions over columns based on user defined threshold value T. These IB queries results give very useful business information to decision makers. These queries can give an opportunity to minimize its processing and execution time [14, 15].

## III. ICEBERG QUERY EVALUATION

The Iceberg Query Evaluation Fong et al firstly proposed probabilistic algorithms and followed by it, hybrid and bucket algorithm are developed for evaluation of IB queries efficiently [16, 17 ]. The column oriented databases are growing vertically to accommodate required space for user data, whereas row oriented databases are arranging the data in row wise [6, 7]. The column oriented databases are more powerful over row oriented databases.

The sampling and course counting algorithms are basic building blocks for evaluation of IB queries and the query size is measured in order to predict IB results. This reduces their query execution time and memory size [9] but the results produced *false positives and false negatives*. To avoid the problems in above algorithms, hybrid strategies have been developed to achieve optimized query execution.

Bin He *et al* [16] proposed a method to evaluate IB queries by reducing CPU and memory requirements greatly by using compressed bitmap indexing technique. A bitmap for an attribute is represented as *m x n* matrix. The *m* is a row that represents number of tuples in the database and *n* is the number of columns that denotes the different attribute values. The presence of an attribute in the database is represented as 1 and 0 otherwise. The bitmap original vectors are stored in the existing free memory by using compressed bitmap indexing technique.

The partition algorithms, Jinuk Bae[18] *et a*l are proposed a partition algorithm that selects the candidates by logically partitioning the given relation R and delay the partitioning to use more efficiently and buckets are filled with all useful candidates .It is observed that this algorithm is suffering from memory size, data order and data distribution.

## IV. THE RANDOM BIT OPERATION (RBO) ALGORITHM

This algorithm takes two bitmap vectors according to count of 1 Bit values and priority Queue is created with help of the vectors. Based on the priority value, two vectors are picked up and performed AND operation between them. Algorithm: Random 1bit operation in the BMP vector.

Input: Random1bitoperator (BMP vector *BMP1*, BMP vector *BMP2*, pos *n)*
Output: IB_Results
Step 1: Select *BMP1, BMP2*
Step 2: Select *n*
Step 3: Scan for *n* bits on *BMP1, BMP2*
Step 4: Set *n* value
Step 5: Execute AND operation

$R_t$ ⟶ *BMP1 AND BMP2: n*
Step 6: Reassign *BMP1 and BMP2*
*BMP1* ⟶ *BMP1 XOR* $R_t$ ; *BMP2* ⟶ *BMP2 XOR* $R_t$
Step 7: if (*Rt_1* bitcount > *W1*)
$R_t$ ⟶ IB _Results
Step 8: End

**Figure 4.1: Algorithm for Random 1 bit calculation**

The proposed method also uses a bitwise AND action between the BMP vectors. The values of the BMP vector are also changed by performing XOR operation on the BMP vectors with the result of AND operation.

$R_t = P\ AND\ Q;$ $P = P\ MINUS\ R_t$ ; $Q = Q\ MINUS\ R_t$

In above, $R_t$ is a resultant vector values and P and Q Bitmap vectors. As specified above, the method uses a random 1 bit operation before performing AND operation on Bitmap vectors. The Bitmap vectors are retrieved from the above method. Firstly, pick the *n* arbitrary 1 bit from the two vectors, the *W1* contains n values which are related bitmap vectors (threshold value) and the count 1 bit threshold. But, if *W1* 1 bit count is greater than many vectors, those vectors whose 1bit count is less can be rejected from BMP vectors. No AND operation is performed on them. Relevant vectors are the vectors towards the random assignment are possible in order to perform AND operations.W1 is the minimum n random value, since if the vectors that are not in the range of W1are not considered as relevant vectors. These irrelevant vectors are pruned from BMP indices. Consider the example

$P \Rightarrow$

$p_1$ : [01011]

p2:[1010011001]

p3 : [0000000110]

$Q\ Q \Rightarrow$

$q_{1[}$10010010]

q2 : [
q2[11001]

$q_3$ : [ $q_3$ : [000110]

Priority queue is produced as ,

| $P$ | $Q$ |
|---|---|
| $p1$ :[1010011001] | q1: [10010010]<br>$q_2$ : [ $q_2$ : [11001] |
| $p2$ :[01011] | |
| $p_3$ :[0000000110] | 0110 $q_3$ : [000110] |
| $W_1$ = 2; | |
| | $Q \Rightarrow$ |
| $p_2$ $_{P1}$ :[1010011010] | $q_1$: [0010010010] |
| $P$ $p_2$:[01011] | $q_2$ : [11001] |
| $p_3$ $_{P3}$ :[0000000101] | $q_3$ : [000110] |

In the above example $p_3$ and $q_3$ are removed from the BMP indices, since they are not satisfying condition specified as $W_1$. The priority queue $p_2$ and $q_1$ are measured as MSB vectors and primarily AND operation is performed on those two vectors. In above, 3 is the random n bit value for the vectors $p_1$ and q1. Therefore AND operation can be represented as below,

**Table 4.1: Attributes with their corresponding bit positions**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *x2* : | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| *y1* : | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| *R* : | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

If $R_t$ has 1 bit count greater than the $W_1$, then it is added to the IB results. The $R_t$ is used for performing XOR operation on both *P a*nd Q attributes. If several vectors possess 1 bit count than the T after XOR operation, then that vector is added to the BMP table. The AND operation is performed again on the resultant Table .The same process is repeated until any one of the vector become empty. By the observation from the above example it can be stated that, with use of random 1bit calculation method the AND operations are drastically reduced.

## V. CONCLUSION

In this research work, we presented an algorithm to evaluate iceberg queries using bitmap indices. The proposed approach is dynamic in nature and flexible enough to reduce the vector values with respect to dimensions. The outcome of the current research is the efficient Iceberg Query Evaluation algorithm that is capable of handling massive datasets. Empirical results imply the practical benefits of the proposed algorithm are giving better performance in evaluating iceberg queries.

## REFERENCES


[1] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D.Ullman, "Computing IB Queries Efficiently". In Gupta et al. [GSW98], pp. 299-310.

[2] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology", *ACM SIGMOD Record*, 26(1):65–74, March 1997.

[3] J. Han, J. Pei, G. Dong, K. Wang, "Efficient Computation of IB Cubes with Complex Measures",*SIGMOD'01*

[4] K.Ross and D. Srivastava, "Fast computation of sparse datacubes", *VLDB'97*.

[5] K.S. Beyer and R. Ramakrishna, "Bottom-Up Computation of Sparse and IB CUBES", *Proceedings of ACM SIGMOD International Conference Management of Data*, pp.359-370, 1999.

[6] Papadis, D, Kalnis P, Zhang, J and Tao Y "Efficient OLAP Operations in spatial data warehouses", *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, Springer, pp443–459, 2001.

[7] S. Chaudhuri and L. Gravano, "Evaluating Top-k Selection Queries". In Atkinson et al. [AOV+99], pp. 397-410.

[8] F. Deiege and T.B. Pedersen, "Position List Word Aligned Hybride: Optimizing Space and Performance for Compressed BMPs", *Proceedings of International Conference Extending Database Technology (EDBT)*, pp. 228- 239, 2010.

[9] Amr El-Helw, Kenneth A.Ross, Bishwaranjan Bhattacharjee, ChristianA. Lang, and George Mihaila, "Column-oriented query processing for row stores", In Proceedings of the International Workshop on Data Warehousing and OLAP, pages 67–74, 2011.

[10] H.E. Blok, "Database Optimization Aspects for Information Retrieval", Ph.D. thesis, University of Twente, Enschede, The Netherlands, April 2002, ISBN 903651732X.

[11] R.S. Choenni, M.L. Kersten, A. Saad, and J. van den Akker, "A Framework for Multi-Query Optimization", Report CS-R9638,CWI, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 1996.

[12] M.-C. Wu and A. P. Buchmann, "Encoded BMP indexing for Data Warehouses". In *Proceedings of ICDE 1998*, pages 220–230, IEEE Computer Society, 1998.

[13] P. Flajolet and G.N. Martin,"Probabilistic counting algorithms for database applications". Journal of Computer System Sciences, 31:182-209, 1985.

[14] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusam, R. (Eds.), Advances in Knowledge Discovery and Data Mining", The AAAI Press/the MIT Press, 1996.

[15] K.Stockinger, D. Duellmann, W. Hoschek, and E. Schikuta, "Improving the performance of high energy physics analysis through BMP indices". *In Proceedings of DEXA 2000*, September 2000.

[16] K.-L. Wu and P. Yu, "Range-based BMP indexing for high cardinality attributes with skew", Technical Report RC 20449, Watson Research Division, Yorktown Heights, New York, May 1996.

[17] P.E.O'Neil and D.Quass, "Improved Query Performance with Variant Indexes". *In SIGMOD Conference*, pp.38–49, 1997.

[18] Jinuk Bae and Sukho Lee, "Partitioning Algorithms for the Computation of Average IB Queries", Springer-Verlag, ISBN: 3-540-67980-4, pp.276 – 286, 2000.

**[1]Mr.M.Laxmaiah** is a Research Scholar in Jawaharlal Nehru Technological University, Kukatpally, Hyderabad. He is currently working as Professor of Department of Computer Science and Engineering in CMR Engineering College, Kandlakoya (V), Medchal (M), Hyderabad, Telangana, India. He has 18 Years of experience in Teaching and 5 Years of experience in Research field. He has 12 research Publications at International Journals and Conferences. His areas of interest include Database Management Systems, Cloud Computing, Compiler Design and Data warehousing & Data Mining.

**[2]Dr.A.Govardhan** did his BE in Computer Science and Engineering from Osmania University College of Engineering, Hyderabad in 1992.M.Tech from Jawaharlal Nehru University, Delhi in 1994 and PhD from Jawaharlal Nehru Technological University, Hyderabad in 2003.He is presently Director of School of Information Technology, JNTUH, Kukatpally, Hyderabad. He has awarded many awards. He has guided more than 130 M.Tech projects and number of MCA and B.Tech projects. He has 200+ research publications at International / National Journals and Conferences. His areas of interest include Databases, Data Warehousing &Data Mining, Information Retrieval, Computer Networks, Image processing and Object Oriented Technologies.