An Algorithm to Evaluate Iceberg Query using Compacted Bitmap Vector

V. Shankar
Associate Professor
Faculty of Computer Science and Engineering
Kakatiya Institute of Technology & Science (KITS),
Warangal, Andhra Pradesh, India - 506015

C.V.Guru Rao, PhD.
Professor & Head
Department of Computer Science and Engineering
S.R.Engineering College
Warangal, Andhra Pradesh, India - 506371

ABSTRACT
The data storing and retrieving are playing a major role in the data clustering and data warehousing techniques. The effectiveness of a data retrieving method depends upon the data specific queries for retrieving the data from the database. Iceberg query is a unique class of aggregation query, which computes aggregate values above a given threshold. Many data mining queries are basically ice berg queries. The major part taken into the consideration about the AND operation in the iceberg queries. The reduced number of AND operation increases the effectiveness of the iceberg query. In this work, an efficient iceberg query evaluation process is proposed by reducing the bitwise AND operations needed to find the item pairs. In the proposed approach two solutions are introduced to reduce the bitwise AND operations. Randomly identifying 'N' 1-bit positions instead of first 1-bit position and reducing the zero bit values from the Most Significant Side so that the bit map vector will be reduced in such a way, the bitwise operations needed is reduced. The experimentation is conducted on two datasets in order to evaluate the performance of the proposed iceberg query evaluation algorithm. In the case of retail datasets, the time required for processing 100000 tuples is 864 milliseconds. On the other hand, the same for synthetic dataset, T10I4D100K, is 910 milliseconds.

Keywords
Database, iceberg query, bitwise-AND operation, dynamic pruning, bitmap table.

1. INTRODUCTION
The relational database systems nowadays like DB2, Oracle, SQL Server, Sybase, MySQL, PostgreSQL, and column oriented databases Vertica, MonetDB, LucidDBare all using general aggregation algorithms [4][5][6]to answer iceberg queries. Many database applications, ranging from decision support to information retrieval, involve SQL queries that compute aggregate functions over a set of grouped attributes and retain in the result only those groups whose aggregate values satisfy a simple comparison predicate with respect to a user-specified threshold [3]. Many practical applications, including Data Warehousing [1], Market-basket Analysis [2] and Information Retrieval [7], rely on Iceberg queries. Such queries compute aggregate functions over a set of attributes and return the aggregate values which are above some threshold. They are called Iceberg queries because the result is usually very small (i.e., the tip of an Iceberg) compared to the input set. Iceberg query [8] is a unique class of aggregation query, which computes aggregate values above a given threshold. Many data mining queries are basically ice berg queries. For instance, market analysts per form market basket queries on large data warehouses that store customer sales transactions. These queries discover user buying patterns, by finding item pairs that are brought together by many customers. Target sets are item-pairs, and 'T' is the minimum number of transactions required to support the item pair. Since these queries operate on very large datasets, solving such iceberg queries efficiently is an important problem.

The name *Iceberg* query is coined by Fang *et. al.* in [9].An iceberg query is used for extracting data from a particular database under some specified conditional parameter. Consider the following query,

$$SELECT\ x, y$$
$$COUNT(*)\ FROM\ R$$
$$GROUP\ BY\ COUNT(*) >= 2$$

The above listed query is assigned to fetch the count of x and y from the table R under a parameter, which specifies the count of x, y greater than 2 should be selected. The above listed types of queries are called iceberg queries. The use of iceberg queries reduces the time for fetching data from a large database. The main advantage is the aggregate function used with the iceberg queries gives more weightage to the iceberg queries. The use of iceberg queries becomes so frequent in new data management system, because of their precise data extraction in a limited time. Thus lots of recent researches are concentrating on improving the efficiency of iceberg queries.

Even with the specific feature iceberg queries are not utilized completely in large database, with the threshold constraint, an iceberg query usually only returns a very small percentage of distinct groups as the output, which resembles the tip of an iceberg. Because of the small result set, icebergs queries can potentially be answered quickly even over a very large data set. However, current database systems and/or approaches do not fully take advantage of this feature of iceberg query [10]. Database systems currently do not employ special techniques to process iceberg queries operating on large databases. That is, independent of the threshold value, they typically use the different approaches like [3], Sort-Merge-Aggregate (SMA),the relation is completely sorted on disk with regard to the group-by attributes and then, in a single sequential scan of the sorted database, those groups whose aggregate values meet the threshold requirement are Output, and Hybrid-Hash-Aggregate (HHA) ,the relation is recursively partitioned using hash functions, resulting in partitions in which the distinct groups in the available main memory where they are subsequently processed.

In accordance with the above mentioned details, a method is needed to evaluate the iceberg queries. In this paper, an approach for evaluation of iceberg query is plotted according to two methods. The methods, illustrated in the paper are a method of adaptive pruning and a method called random 1 bit calculator. The adaptive pruning step is concentrated on reducing the number of zero bits in the bitmap vector corresponding to the attributes. The adaptive pruning helps in reducing the number of AND operations, which helps in the

improvement of iceberg queries. The second method concentrates on executing the AND operations by randomly selecting a number of 1 bits from the attributes. The random selection of 1 bit is based on a threshold called count 1 bit threshold, which characterizes how much 1 bit should be possessed by a bitmap vector. The iceberg results are also taken on the basis of the threshold set for the 1 bit positions.

The main contributions of the paper are,

- An adaptive pruning method is used to reduce the zero bits in the vector.
- A random 1 bit operator is used to fetch the iceberg results

The rest of the paper is organized as, the section 2 gives a review of some related works regarding evaluation of iceberg quries. Section 3 contains Motivational algorithms behind this research. 4th section gives details of the proposed approach with mathematical models. 5th section gives the results and discussion about the proposed approach and with the 6th section we conclude our research work.

2. REVIEW OF RELATED WORKS

A handful of researches are available in literature for iceberg queries. In recent times, the evaluation of iceberg queries in distributed manner has attracted researchers significantly due to the demand of scalability and efficiency. Here, we review the recent works available in the literature for evaluation of iceberg queries. Researchers were always very keen to find out the efficient ways to execute the iceberg queries because of the limited computing resources and the large data over which queries are executed. There are results which are showing that executing iceberg queries on data takes more time than finding the association rule from the data sets .Therefore scientist involved in domain of data warehousing, information retrieval ,knowledge discovery are continuously working on the problem iceberg query execution .Many novel ideas and techniques have been proposed by number of researchers. In this section we will list some of those methodologies recently appeared in the literature

Recently, Hsiao H *et al* [11].of IBM Almaden Research Center, San Jose proposed a approach of executing the iceberg queries efficiently using the compressed bitmap index. Bitmap index, which builds one bitmap vector for each attribute value, is gaining popularity in both column-oriented and row-oriented databases in recent years. In that paper, they exploited the property of bitmap index and developed a very effective bitmap pruning strategy for processing iceberg queries. There index-pruning based approach eliminates the need of scanning and processing the entire data set (table) and thus speeds up the iceberg query processing significantly. Experiments show that their approach is much more efficient than existing algorithms commonly used in row-oriented and column-oriented databases.

JinukBae*et al*[12].in their paper Partitioning Algorithms for computation of Average Iceberg Queries introduce the theorem to select candidates by means of partitioning, and propose POP algorithm based on it. The characteristics of this algorithm are to partition a relation logically and to postpone partitioning to use memory efficiently until all buckets are occupied with candidates. Experiments show that proposed algorithm is affected by memory size, data order, and the distribution of data set.

Bin He*et al*[10]Decision support and knowledge discovery systems often compute aggregate values of interesting attributes by processing a huge amount of data in very large databases and/or warehouses. In particular, *iceberg query* is a special type of aggregation query that computes aggregate values above a user provided threshold. Usually only a small number of results will satisfy the threshold constraint. Yet, the results often carry very important and valuable business insights. Because of the small result set, iceberg queries offer many opportunities for deep query optimization. However, most existing iceberg query processing algorithms do not take advantage of the small-result-set property and rely heavily on the *tuple-scan based* approach. This incurs intensive disk accesses and computation, resulting in long processing time especially when data size is large. Bitmap index, which builds one bitmap vector for each attribute value, is gaining popularity in both column-oriented and row-oriented databases in recent years.

Alfredo Cuzzocrea et al [14] devised an algorithm which employed a probabilistic framework to prevent cells which are unlikely to satisfy the iceberg condition from being computed. The output of our algorithm was an incomplete iceberg cube, which was efficiently computed and prone to be refined, in the sense that the user can decide to go through the computation of the cells which were estimated irrelevant during the previous invocations of the algorithm. Fianny Ming-fei Jiang [15] have introduced the IX-cube (Iceberg XML cube) over XML data to tackle the problem. They extended OLAP operations to XML data. They also developed efficient approaches to IX-Cube computation and OLAP query evaluation using IX-cubes. Stockinger, Kurt. [16] have presented a strategy to efficiently answer joint queries on both types of data. By using an efficient compression algorithm, their compressed bitmap indexes, called FastBit, were compact even when they contain millions of bitmaps.

3. MOTIVATION BEHIND THE APPROACH

The data storing and retrieving are playing a major role in the data clustering and data warehousing techniques. The effectiveness of a data retrieving method depends upon the accuracy level of the method in limited or less amount of time. In such methods retrieving data from defined database is done with the help of database queries. Similarly, Bin He*et al* [10] proposed an evaluation of iceberg queries. The approach is an evaluation of iceberg query using compressed bitmap index. The main processes which are used in the efficient iceberg query evaluation are compressed bitmap and dynamic pruning of the bitmap index. The highlighted feature in Bin He*et al* [10] approach is a specialized vector alignment of the bitmap indices, which deals with AND operation between distinct values in attributes specified by the iceberg query. The approach produces iceberg results with less AND operations and that is a specification provided by the specified vector alignment. Inspired from the research, in this paper an efficient iceberg query extraction is done with an adaptive pruning technique and random 1 bit value vector.

4. PROPOSED ICEBERG EVALUATION APPROACH

In this section, an efficient iceberg query evaluation is proposed inspired from the evaluation of iceberg queries stated in [1]. The proposed approach concentrating on evaluation of iceberg queries with bitmap indices of the attributes quoted in the iceberg query. The bitmap is processed with an adaptive pruning technique to make the bitmap indices easier to process. An improved vector alignment with priority queue is subjected to the effective evaluation iceberg query. The proposed approach is defined over three phases, initially a preprocessing is done to extract the bitmap indices. The second phase deals

with the adaptive pruning of the bitmap indices and the third is the specialized vector alignment to extract the iceberg results. An Iceberg query can be defined as a special class of aggregation query, which computes aggregate values above a given threshold. It is of special interest to the users, as high frequency events or high aggregate values often carry more important information. In the current approach, iceberg queries with aggregation functions having the anti-monotone property are considered. An example of the iceberg query is shown below,

$$SELECT < attributes > AGG(*)\ FROM < database >$$
$$GROUP\ BY\ AGG(*) >= < condition >$$

Here, x and y are attributes and AGG shows aggregate functions such as COUNT, SUM, MAX, etc. and the comparison predicate used here is "greater than or equal to (>=) ". The proposed approach uses the following iceberg query as an example,

Example,

$$SELECT\ x, y\ COUNT(*)\ FROM\ R\ GROUP\ BY\ COUNT(*) >= 2 \quad (1)$$

Preprocessing database (bitmap indexing)

Bitmap indices are commonly used in databases, especially for data warehousing applications and in column stores. A bitmap for an attribute (column) of a table can be viewed as a v × r matrix, where v is the number of distinct values of the column and r is the number of tuples in the table. Each value in the column corresponds to a bitmap vector of length r, in which, the k^{th} position of the vector is 1 if this value appears in the k^{th} row, and 0 otherwise. In the case of proposed approach, the distinct values of the attribute are taken into the consideration. If the position belongs to a particular value, the presence is marked by 1 and the absence is marked by 0. Each of the attribute has individual bitmap vector for each of the value corresponding it. Following illustrations describes the bitmap indices.

	Position vector								
	0	1	2	3	4	5	…	…	n
A1	a1	a2	a1	a3	a1	a2	…	…	an
A2	b1	b2	b3	b2	b1	b5	…	…	bn
…									
An	…	…	…	…	…	…	…	…	…

Fig.1. Attributes and distinct value

According to the attributes and the position of their distinct values, bitmap vector is constructed for every distinct value in all the attributes. Consider the bitmap vector generated for the attribute A1.

$$A1 \Rightarrow a1 : [101010.....0],$$
$$a2 : [010001.....0], a3 : [000100.....0], ... an : [000000....1]$$

Similarly, bitmap vectors for all other attributes are calculated. Continuing with iceberg query presented in the equation 1, the bitmap vector is explained in detail by considering X,Y as attributes and their distinct values as,

$$X : [x1, x2, x3] \qquad Y : [y1, y2, y3]$$

	Position vector							
	0	1	2	3	4	5	6	7
X	x1	x2	x1	x3	x1	x2	x3	x2
Y	y1	y2	y3	y3	y1	y3	y2	y3

Fig.2. data table

$$X \Rightarrow \qquad Y \Rightarrow$$
$$x1 : [10101000] \qquad y1 : [10001000]$$
$$x2 : [01000101] \qquad y2 : [01000110]$$
$$x3 : [00010010] \qquad y3 : [00010011]$$

Fig.3. Bitmap indices

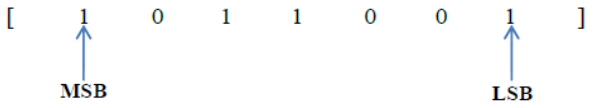
The proposed approach, the bitmap is generated using the compressed equality encoding []. The uncompressed bitmap generation method is avoided, because it uses large amount data space and increase the time complexity.

Adaptive dynamic pruning of bitmap vectors

This phase of the proposed approach deals with pruning of the bitmap vectors for obtaining effective results for the iceberg queries. When an iceberg query is initiated, the distinct values of the attributes specified in the iceberg query is subjected for bitwise AND operation. The bitwise AND operation is extremely time consuming, such as, if there are a million values is there in each of the bitmap vectors then the AND operation needed for two vectors is a million and which is quite a tedious process. So the pruning is done in order to reduce the number of AND operations.

Consider the iceberg query illustrated in equation 1 and their bitmap indices from fig. 3. In the case, attributes X and Y has distinct values *[x1,x2,x3]* and *[y1,y2,y3]* respectively. According to the iceberg query, the AND operation is executed between*[x1:y1, x1:y2, x1:y3, x2:y1, x2:y2, x2:y3, x3:y1, x3:y2, x3:y3]*. The count of 1s in the resulting bitmap vector is selected and added to the iceberg results according to a threshold value. This method is time consuming because of unwanted empty AND operations such as AND operation between 0 bits. The proposed approach uses two threshold values for reducing the AND operation. The adaptive pruning method using in the proposed approach is a 0 bit pruning method. The proposed approach consider significant bits in a bitmap vector, a leasy significant bit (LSB) and most significant bit (MSB).

Example;



The MSB is assigned to the first 1 bit value in the bitmap vector. The LSB is assigned according to a ***count zero threshold*** (Z_0), which is a threshold set for the count of zeros after a 1 bit value. If the zero count after a 1 bit value is higher than Z_0, the 1 bit is set as LSB of the bitmap vector. The bit values come after the LSB is pruned. In this way, a number of zero bit are erased, which results in less number of AND operations.

Example;

The Z_0 is not a randomly assigned value, it is assigned according to the count 1 bit threshold (Z_1), which is the count of 1 bit in the bitmap vector. The Z_1 value is deciding parameter for the Z_0 value. This feature of the bit pruning, make it as an adaptive pruning method. When compared to earlier methods for pruning, the adaptive pruning method has advantage. The adaptive pruning method reduces the time by eliminating zero bit value. The elimination of the zero values also helps in reducing the complexity of the process, since that much AND operations are reduced. By considering the following example, the adaptive pruning technique can explained clearly,

Bitmap vectors,

$$BM1 \rightarrow [\ 1\ 0\ 0\ 1\ 1\ 0\ 0]$$
$$BM2 \rightarrow [\ 1\ 1\ 0\ 0\ 1\ 1\ 1]$$
$$BM3 \rightarrow [\ 1\ 0\ 1\ 0\ 0\ 0\ 1]$$

In the above example, *BM1*, *BM2* and *BM3* are three bitmap vectors. The Z_1 threshold is set as 3 and according to that the Z_0 threshold is set as 2. I.e. the number consecutive zeros, if two or more consecutive zeros are occurred, then the bits after that will be pruned from the LSB.

Bitmap vectors after pruning,

$$BM1 \rightarrow [\ 1\ 0\ 0\ 1\ 1\ ]$$
$$BM2 \rightarrow [\ 1\ 1\ 0\ 0\ 1\ 1\ 1]$$
$$BM3 \rightarrow [\ 1\ 0\ 1\ ]$$

In the case of BM1 after the MSB there is two consecutive is zeros, but according to the proposed approach, the pruning is executed only after the LSB is set. Thus, the two zeros are neglected and the next one bit is set as LSB. When two consecutive zeros are again occurred, the pruning is subjected. The case of BM2 is that, it has consecutive zeros but the density of the one is higher in that vector, thus no pruning is applied. In the case of BM3, the LSB is set in the early bit and after that the Z_0 condition is satisfied. So pruning is applied and after the pruning it does not satisfy the Z1 conditions thus it is rejected from the bitmap vectors.

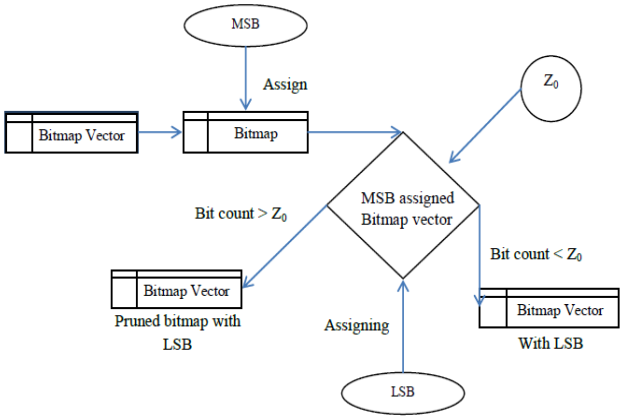


Fig.4.Adaptive pruning

Random 1 bit Calculation

The adaptive pruning is an initial measure to reduce the number of AND operations between bitmap vectors in the bitmap indices. To make the AND operation more precise the earlier approach uses a vector alignment method, in which the first 1 bit position is selected from the bitmap vectors which are selected for the AND operation. A priority queue is set for the bitmap vectors, which contains the most significant vectors in the top of the queue. On disadvantage of this method is that, if the count of 1 bit in both vector are same and first 1 bit position are not close then AND operation produces harder results and also consumes more time for scanning the first 1 bit position. In order to avoid this problem to an extent, the proposed approach uses a random 1 bit process.

In similar to the earlier approach the proposed approach also uses a bitwise AND operation between the bitmap vectors. The values of the bitmap vector are also changed by XORing the bitmap vectors with the result of AND operation.

$$R = X \cap Y; \qquad X = X \otimes R; \qquad Y = Y \otimes R$$

Here R represents the result of the bitwise AND operation and X and Y are the bitmap vectors used for the AND operation. As mentioned above, the proposed approach uses a random 1 bit operation on the bitmap vectors before the AND operation. The bitmap vectors are obtained from the adaptive pruning technique. Initially, select the n random 1 bit from both vectors, the n values should be related to the Z_1, the count 1 bit threshold. If there are any vectors with 1 bit count less than Z_1, that vector is rejected. The vectors, in which the random assignment are possible, is called a relevant vectors for the AND operation. The minimum value of the random n value is Z_1, because those vectors which are not in the range of Z_1 are considered as irrelevant vectors and such vectors can be pruned from the bitmap indices. Consider the following example,

$X \Rightarrow$	$Y \Rightarrow$
$x1:[01011]$	$y1:[10010010]$
$x2:[1010011001]$	$y2:[11001]$
$x3:[0000000110]$	$y3:[000110]$

Priority queue is constructed as,

$X \Rightarrow$	$Y \Rightarrow$
$x2:[1010011001]$	$y1:[10010010]$
$x1:[01011]$	$y2:[11001]$
$x3:[0000000110]$	$y3:[000110]$

$Z_1 = 2$;

$X \Rightarrow$	$Y \Rightarrow$
$x2$:[1010011010]	$y1$:[0010010010]
$x1$:[01011]	$y2$:[11001]
$x3$:[0000000101]	$y3$:[000110]

Here *x3* and y3 are pruned from the bitmap index, because they do not satisfy the conditional parameters such as Z_1. According to the priority queue x2 and y1 are considered as the most significant vectors and initial AND operation is assigned on those two vectors. For vectors x2 and y1 the random n bit value is set as 3. Thus AND operation can be illustrated as,

$x2$:	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 has 1 bit count greater than the Z_1, then it is added to the iceberg results. The R is selected for XORing with the attributes X and Y, if any vector possess 1 bit count higher than the threshold after XOR operation, that vector added to the bitmap table again for AND operations. The above listed process continues till any vector got empty. From the example it can be stated that, with the use of random 1 bit process the number of AND operations are further reduced.

Algorithm: Random 1bit operation in the bitmap vector
Random1bitoperator (bitmap vector *v*, position *n*)

Step1: **Select***v1, v2*
Step2: **select***n*
Step3: **Scan***n* bits on *v1, v2*
Step4: **set***n* value
Step5: **Execute** AND operation,
$R \rightarrow v1 \cap v2 : n$
Step6: **reassign** v1 and v2,
$v1 \rightarrow v1 \otimes R ; v2 \rightarrow v2 \otimes R$
Step7: if (R_1 bitcount> Z_1)
$R \rightarrow iceberg\_results$
Step8: **End**

Pseudocode.1.Random 1 bit operation

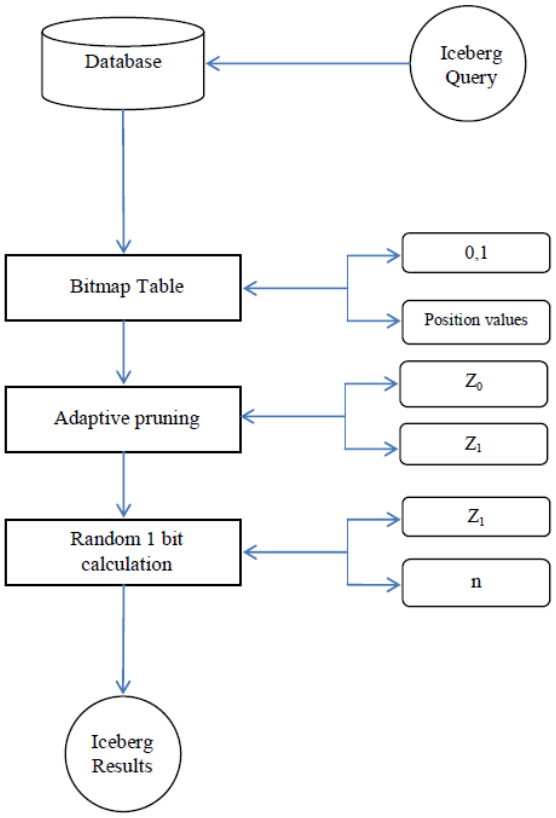


Fig.5.Effective Iceberg evaluation: Block diagram

The above fig. 5 shows the architecture of the of the proposed iceberg evaluation process. The architecture shows three phases in the process and their attributes. The attributes are the values, which are used for the working of a particular phase. The processing of an iceberg query to iceberg result is shown in the architecture.

5. EXPERIMENTAL RESULTS

The proposed approach deals with the clustering of data based on the ABC algorithm. The method we proposed incorporates the FCM function with ABC algorithm for getting better efficiency. The performance of the proposed approach is evaluated in the following section under different evaluation criteria. All algorithms are implemented in JAVA language and executed on a core i5 processor, 2.1MHZ, 4 GB RAM computer.

Dataset description

The proposed approach uses two types of dataset for evaluating the iceberg results. The data sets used are a synthetic data and real data. The datasets are extracted from the Frequent Item set Mining Dataset Repository [13]. The synthetic data used for experimental evaluation is T10I4D100K, which were generated using the generator from the IBM Almaden Quest research group. The second dataset that is used in the proposed approach is the RETAIL dataset. The retail dataset was generated by Tom Brijs and which consists of retail market basket data from a Belgian retail store.

The performance evaluation of the proposed approach is based on the number of tuples and the threshold values. The tuples represent the number elements, which an attribute consist or the number of distinct value possessed by the attributes. The threshold value mentioned here is referred to the iceberg threshold. The main evaluation factor that is considered here is the time for execution.

Performance evaluation

The performance evaluation of the proposed approach, mainly deals with less time for and operations. Thus the time of execution is concentrated more on the approach. The two dataset are tested with the proposed iceberg evaluation algorithm. The responses of the datasets according to the proposed approach are detailed in the following graph. The experimentation is conducted on different number of tuples in the dataset. The responses of the proposed iceberg evaluation are different for different number of tuples.

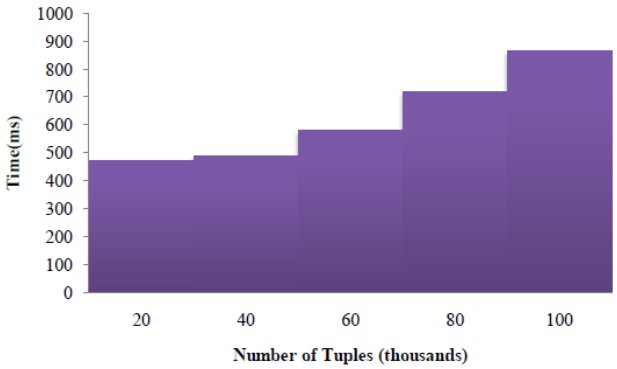


Fig.6. Time based evaluation of Retail dataset

The fig.6 shows the responses of the retail dataset to the proposed iceberg evaluation algorithm. The retail data is evaluated by dividing 5 partitions with uniformly increasing number of tuples. The response of time to initial partition is comparable less than the other partitions. Then, as the number of tuples increases the time for execution is also increases proportionally. As the number of tuples reaches to a high level the time of execution is comparable high.

Fig.7. Time based evaluation on Synthetic dataset

The fig.7 represents response of the synthetic dataset according to the proposed iceberg evaluation algorithm. The response of the synthetic data is similar to the retail dataset, as it is also respond to the time proportionally when the number of tuples increases. Since, the size of the data is different the time of execution value is little changed. The time required for execution of 100000 data is 910 mille seconds. In the case retail data set the time required for executing 100000 tuples with the proposed iceberg algorithm is 864 mille seconds.

The experimentation is extended to evaluate the sharpness of the proposed iceberg query evaluation algorithm. The iceberg threshold has been changed to different values to evaluate the input datasets. The datasets responses are taken according to four different values of the iceberg threshold. The threshold value is set according to the count of the attributes since the icberg query is subjected for the "COUNT" operation.

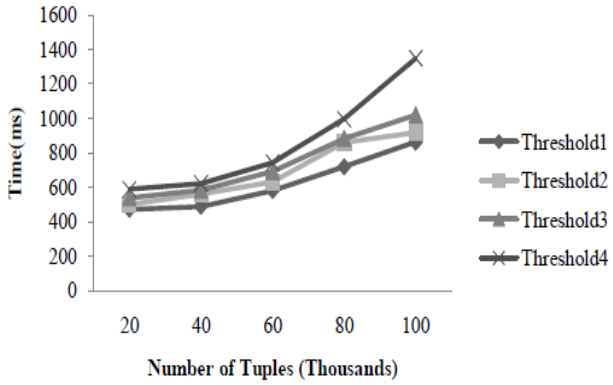


Fig.8. responses of Retail dataset

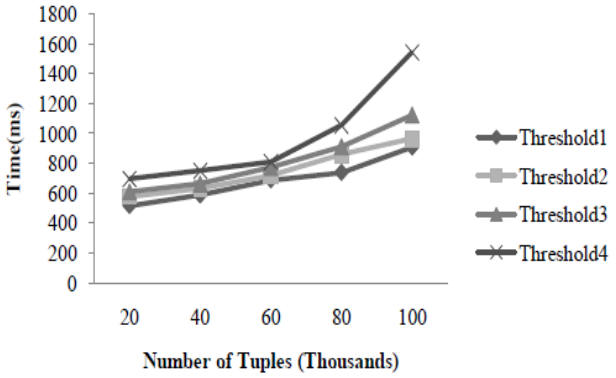


Fig.9. Responses of synthetic dataset

Fig.8 is the graph plotted according to the responses from the retail dataset for different threshold values. The analysis from the graph shows that, as the threshold increases the time for execution of the data also increases. This happens because as threshold increases, the algorithm searches for most possible AND operations within the limit of the threshold. The above plotted result is based on the COUNT operation in the database. The fig.9 represents the responses of the T10I4D100K dataset according to the different threshold values.

6. COMPARATIVE ANALYSIS

This section includes the comparative study of performance of the proposed iceberg evaluation algorithm and an existing iceberg evaluation method. The existing method refers to the efficient iceberg query evaluation using compressed bitmap index by Bin He*et al*. [1]. The comparison study is based on the responses of retail dataset with two different algorithms. Both the algorithms are based on the bitmap index table and the AND operation between the attributes. The aim of the algorithms is to reduce the execution time by reducing the unwanted AND operation.

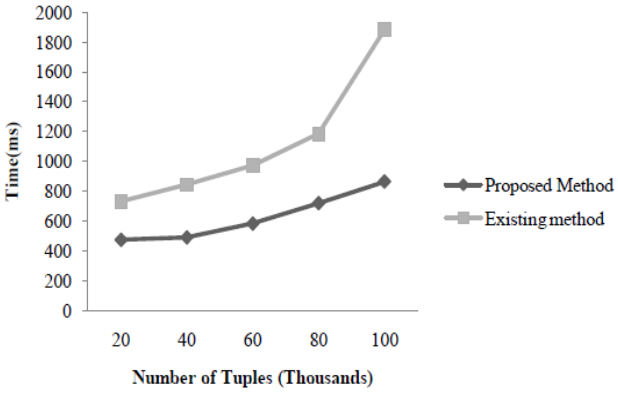


Fig.10. Comparative study

The fig.10 shows the comparative analysis of the proposed iceberg evaluation algorithm and the existing iceberg evaluation algorithm. The analysis shows that time required for execution of 100000 tuples by the proposed approach are 864 milliseconds; on the other hand, the same for the existing method is 1884. In both the cases the time is increasing according to the increase of the number of tuples. Thus we can state that the proposed iceberg evaluation has quick response than the existing approach mentioned in [1].

7. CONCLUSION

In the field of information retrieval, the data retrieval from the database is become more time consuming process as the number of data is increasing day by day. Specialized queries are used for retrieving data from the databases. Iceberg query are similar queries which uses aggregate function and conditional clauses to get the data from the databases quickly. In the proposed approach, an approach for iceberg query evaluation is proposed. The proposed iceberg query evaluation algorithm uses bitmap index table for the iceberg evaluation. The algorithm has two methods, one for reducing the number of AND operations by reducing the number zero bits. The second method is to improve the efficiency of AND operations by randomly selecting the 1 bit values. The experimentation is conducted on two datasets in order to evaluate the performance of the proposed iceberg query evaluation algorithm. In the case of retail datasets, the time required for processing 100000 tuples is 864 milliseconds. On the other hand, the same for synthetic dataset, T10I4D100K, is 910 milliseconds. The futuristic enhancement can be applied to the proposed method by processing the priority queues using some learning algorithms. In another direction, the execution time can also be reduces by eliminating large number of unproductive bitwise-AND operations is would be taken up.

REFERENCES

[1] Kevin S. Beyer and Raghu Ramakrishnan "Bottom-up computation of sparse and iceberg cubes". In Proc. of the Int. Conf. on Management of Data (ACM SIGMOD), pages 359-370, 1999.

[2] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur.Dynamic itemset counting and implication rules for market basket data. In Proc. of the Int. Conf. on Management of Data (ACM SIGMOD), pages 255-264, 1997.

[3] Leela, Krishna P and Tolani, Pankaj M and Haritsa, Jayant R, "On Incorporating Iceberg Queries in Query Processors.", In: 9th International Conference on Database Systems for Advanced Applications:DASFAA, vol.2973, pp.431-442, 2004.

[4] G. Graefe. "Query Evaluation Techniques for Large Databases", ACM, pp.73–170, 1993.

[5] W. P. Yan and Larson, "Data Reduction through EarlyGrouping", In CASCON, page 74, 1994.

[6] P.-A˚. Larson, "Grouping and Duplicate Elimination: Benefits of Early Aggregation" Technical Report MSR-TR-97-36, Microsoft Research, 1997.

[7] A. Broder, S.C. Glassman and M.S. Manasse, "Syntactic clustering of the web.InProc. of the 6th Int. World Wide Web Conference," 1997.

[8] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. F. Naughton,R. Ramakrishnan, and S. Sarawagi. "On the Computation of Multidimensional Aggregates. In VLDB", pp. 506–521, 1996.

[9] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D.Ullman.Computing Iceberg Queries Efficiently. In VLDB, pages299–310, 1998.

[10] 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, pp.1-2, 2011.

[11] Hsiao H, Liu Z, Huang Y, Chen Y, "Efficient Iceberg Query Evaluation using Compressed Bitmap Index" , Knowledge and Data Engineering, IEEE, Issue: 99, pp.1, 2011.

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

[13] Frequent Item set Mining Dataset Repository http://fimi.ua.ac.be/data/

[14] Alfredo Cuzzocrea, Filippo Furfaro and Giuseppe M. Mazzeo, "A Probabilistic Approach for Computing Approximate Iceberg Cubes", Lecture Notes in Computer Science, vol. 5181, pp. 348-361, 2008.

[15] Fianny Ming-fei Jiang, Jian Pei and Ada Wai-chee Fu, "Ix-cubes: iceberg cubes for data warehousing and olap on xml data", in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 905-908, 2007.

[16] Stockinger, Kurt. (2009). Using Bitmap Index for Joint Queries on Structured and Text Data. Lawrence Berkeley National Laboratory: Lawrence Berkeley National Laboratory. Retrieved from: http://www.escholarship.org/uc/item/0db7c07n