Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Compacted Bitmap Vector Technique to Evaluate Iceberg Queries Efficiently

K.Sunil Kumar,
Assoc.Prof in CSE
Vaageswari College of College, JNT University Hyderabad, India

M.Laxmaiah,
Professor in CSE,
Tirumala College of Engineering, JNT University Hyderabad, India

Dr. C.Sunil Kumar
Professor in CSE
Vaageswari College of Engineering, JNT University Hyderabad, India

Abstract— the data storage and retrieving plays vital role in the data clustering (DC) and data warehousing (DW) procedures. The efficiency of a data retrieving technique depends on specific queries for retrieving the data from the relational database. Iceberg (IB) query is a distinctive class of aggregation query, which computes aggregate values beyond a given threshold (TH). Many data mining (DM) queries are mostly IB queries. The major part taken into the kindness about the AND operation in the IB queries. The condensed number of AND operation increases the efficiency of the IB query. In this effort, a well-organized IB query evaluation process is proposed by reducing the bitwise AND operations needed to find the item pairs (IPs). In the proposed scheme two keys are introduced to reduce the bitwise AND operations. Arbitrarily identifying 'N' 1-bit positions instead of first 1-bit position and dipping the zero bit values from the Most Significant (MS) Side so that the bit map vector (BMV) will be condensed so that, the bitwise operations needed is reduced. The testing is conducted on two datasets (DSs) in order to assess the performance of the proposed IB query evaluation algorithm. In the case of retail datasets, the time required for processing 120000 tuples is 857 milliseconds.

Keywords— Database, iceberg (IB) query, bitwise-AND, dynamic pruning, bitmap vector

I. INTRODUCTION

The relational database systems (RDBMS) today like DB2, Oracle 11i, SQL Server 2005, Sybase, MySQL, and column oriented databases (DBs) like Vertica, MonetDB uses general aggregation algorithms **[4, 5 and 6]** to answer IB queries. Many database applications, ranging from decision support to information retrieval (IR), involve SQL queries that calculate aggregate functions over a set of grouped attributes and hold in the result only those groups whose aggregate values assure a simple comparison predicate with respect to a user-specified TH **[3]**. Many realistic applications, including DW and IR rely on IB queries **[1, 7]**. Such kind of queries calculates aggregate functions over a set of attributes and returns the aggregate values which are above some TH. They are called IB queries because the outcome is usually very small compared to the input set. IB query is a distinctive class of aggregation query, which computes aggregate values above a given TH **[8]**. Many DM queries are basically IB queries. Since these queries operate on very large datasets, solving such IB queries proficiently is a significant problem. An IB query is used for extracting data from a particular database (DB) under some specified conditional parameter **[9]**. Consider the subsequent query,

SELECT a, b COUNT* (*) *FROM R
***GROUP BY COUNT* (*) >= 2**

The above query is to fetch the count of a and b from the table R under a parameter, which specifies the count of a, b > = 2 should be chosen. The use of IB queries decreases the time for fetching data from a large DB. The main benefit is the aggregate function used with the IB queries gives more weightage to the IB queries. The use of IB queries becomes so frequent in new data management system (DMS), because of their accurate data extraction in inadequate time. Thus lots of recent researches are focused on improving the efficiency of IB queries. The specific characteristic of IB queries are not utilized completely in large DB, with the TH restriction, an IB query usually only returns small percentage of distinct groups as output, which look likes the incline of an IB. Because of the small result set, IBs queries can potentially be answered quickly even over a very large DS. However, present DB approaches do not fully take advantage of this feature of IB query **[10]**. DBs at present do not employ special techniques to develop IB queries operating on large DBs. That is, independent of the TH value, they normally use the dissimilar approaches like Sort-Merge-Aggregate (SMA),the relation is completely sorted on disk with view to the group-by attributes and then, in a single sequential scan of the sorted DB, those groups whose aggregate values meet the TH requirement are Output and Hybrid-Hash-Aggregate (HHA) the relation is recursively partitioned using hash functions which results in partitions in which the different groups in the available main memory where they are then developed.
With the above mentioned information, a technique is needed to evaluate the IB queries. In this work, an approach for

evaluation of IB query is planned according to two techniques. The methods, illustrated in the work are a technique of adaptive pruning (AP) and a scheme called random 1 bit calculator. The AP step is concentrated on decreasing the number of '0' bits in the BMV consequent to the attributes. The AP helps in decreasing the number of AND operations, which helps in the development of IB queries. The second technique concentrates on executing the AND operations by arbitrarily selecting a number of 1 bits from the attributes. The random selection of 1 bit is based on a TH called count 1 bit TH, which differentiates how much 1 bit should be possessed by a BMV. The IB results are also taken on the basis of the TH set for the 1 bit positions.

The main contributions of the work are,

- An AP technique is used to reduce the '0' bits in the vector.
- A random 1 bit operator is used to obtain the IB results

II. SURVEY

A various no of investigations are available in survey for evaluation of IB queries. In current times, the appraisal of IB queries in distributed manner has attracted researchers considerably due to the claim of scalability and effectiveness. Researchers were always very eager to find out the efficient ways to execute the IB queries because of the limited computing sources and the large DS over which queries are executed. There are results which are showing that executing IB queries on data takes more time than finding the association rule from the DSs. Consequently Researchers involved in domain of DW, IR, knowledge discovery (KD) are constantly working on the problem IB query execution .Many original ideas and methods have been proposed by number of researchers. In this survey some of those methodologies recently appeared in the literature. Recently, proposed an approach of executing the IB queries professionally using the compressed bitmap index (BMI). It builds one BMV for each attribute value, is gaining fame in both column-oriented and row-oriented DBs in recent years. The property of BMI and developed a very effective BM pruning strategy for processing IB queries. There index-pruning (IP) based approach eliminates the need of scanning and processing the entire DS table and thus paces up the IB query processing considerably. Research shows that their approach is much more competent than existing algorithms commonly used in row-oriented and column-oriented DBs **[11]**. Partitioning Algorithms for computation of Average IB Queries initiate the theorem to select candidates by means of partitioning, and propose POP algorithm based on it **[12]**. The characteristics of this algorithm are to partition a relation logically and to delay the partition to use memory proficiently until all buckets are occupied with candidates. Experiments show that proposed algorithm is exaggerated by memory size, data order, and the distribution of DS. Decision support and KD systems often compute aggregate values of motivating attributes by processing a huge amount of data in very large DBs and/or DWs. In meticulous, *IB query* is a special type of aggregation query that computes aggregate values above a user provided TH.

Usually only a small number of results will satisfy the TH limitation. Yet, the results often carry very important and valuable. Because of the small result set, IB queries offer many opportunities for deep query optimization **[10]**. However, most existing IB query processing algorithms do not take advantage of the small-result-set property and rely greatly on the *tuple-scan based* (TSB) approach. This acquires intensive disk accesses and computation, resulting in long processing time especially when data size is large. BMI, which builds one BMV for each attribute value, is gaining fame in both column-oriented and row-oriented DBs in recent years. 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 DB is done with the help of DB queries. Similarly, one author proposed an evaluation of IB queries. The approach is an evaluation of IB query using compressed BMI. The main processes which are used in the efficient IB query evaluation are compressed BM and dynamic pruning (DP) of the BMI. The highlighted feature is specialized vector alignment of the BM indices, which deals with AND operation among distinct values in attributes specified by the IB query. The approach produces IB results with less AND operations and that is a condition provided by the specified vector alignment. Inspired from the research, in this work an efficient IB query extraction is done with an AP technique and random 1 bit value vector.

III. PROPOSED ICEBERG EVALUATION TECHNIQUE

An efficient IB query evaluation is proposed from the evaluation of IB queries. The proposed scheme concentrates on evaluation of IB queries with BM indices of the attributes quoted in the IB query. The BM is processed with an AP method to make the BM indices easier to practice. An improved vector alignment with priority queue (PQ) is subjected to the effective evaluation IB query. The proposed scheme is defined over three phases, initially a preprocessing is done to extract the BM indices. The second phase deals with the AP of the BM indices and the third phase is the specialized vector alignment to extract the IB results. An IB query can be defined as a special class of aggregation query, which computes aggregate values above a given TH. It is of special awareness to the users, as high aggregate values often carry more vital information. In the current approach, IB queries with aggregation functions having the anti-monotone property are considered.

A. Preprocessing database

BM indices are usually used in DBs, especially for DW applications and in column stores. A BM for an attribute of a table can be viewed as a m × n matrix, where 'm' is the number of distinct values of the column and 'n' is the number of tuples in the table. Each value in the column corresponds to a BM vector of length 'n' in which, the I^{th} position of the vector is 1 if this value appears in the I^{th} row and 0 otherwise. In the case of proposed scheme, the different values of the

attribute are taken into the kindness. 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 BM vector for each of the value corresponding it. Following figures describes the BM 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	…	…	…	…	…	…	…	…	…

Figure 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 Z1.

$$Z1 \rightarrow {}_{C}1 : [101010..... \quad 0], \quad {}_{C}2 : [010001..... \quad 0], \ {}_{C}3 : 0],.....,cn : [000100..... \ [000000..... \quad 1]$$

Similarly, BMVs 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

Figure 2: Data table

X →	***Y*** →
***x*1 : [10101000]**	***y*1 : [10001000]**
***x*2 : [01000101]**	***y*2 : [01000110]**
***x*3 : [00010010]**	***y*3 : [00010011]**

Figure 3: Bitmap indices

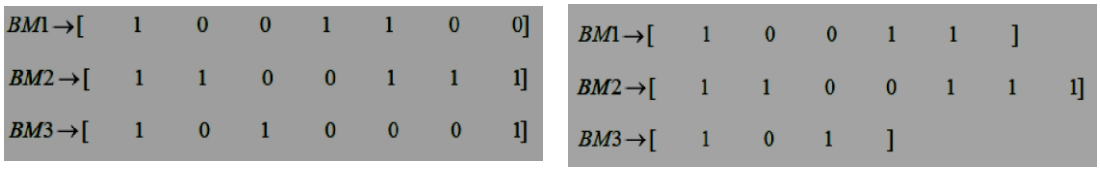
The proposed method, the BM is generated using the compressed equality encoding. The uncompressed BM generation method is avoided, because it uses large amount data space and amplifies the time complexity.

B. Adaptive Dynamic Pruning

This phase of the proposed approach deals with pruning of the BMVs for obtaining effective results for the IB queries. When an IB query is initiated, the different values of the attributes specified in the IB query is subjected for bitwise AND operation. The bitwise AND operation is enormously time consuming, such as, if there are a million values is there in each of the BMVs then they AND operation needed for two vectors is a million and which is quite a tiresome process. So the pruning is done in order to decrease the number of AND operations. Consider the iceberg query illustrated in equation 1 and their bitmap indices shown in the Figure 3. In the case, attributes X and Y has different values *[x1,x2,x3]* and *[y1,y2,y3]* respectively. According to the IB query, the AND function 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 BMV is selected and added to the IB results according to a TH value. This technique is time consuming process because of surplus empty AND operations such as AND operation between 0 bits. The proposed scheme uses two TH values for decreasing the AND operation. The AP method using in the proposed method is a 0 bit pruning method. The proposed method considers important bits in a

BMV, a least significant bit (LSB) and most significant bit (MSB).

The MSB is assigned to the first 1 bit value in the BMV. The LSB is assigned according to a ***count zero TH*** (Z_0), which is a TH 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 BMV. The bit values come after the LSB is pruned. In this method, a number of zero bit are erased, which results in less number of AND operations. 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 BP, make it as an AP method. When compared to earlier methods for pruning, the AP method has advantage which is shown in the Figure 4. The AP method decreases the time by eliminating '0' bit value. The exclusion of the '0' values also helps in decreasing the complexity of the method, since that much AND operations are decreased. By considering the following example, the AP technique can explained clearly BMVs,



In the above example, *BM1*, *BM2* and *BM3* are three BMVs. The Z_1 threshold is set as 3 and according to that the Z_0 TH 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. BMVs after pruning, In the case of BM1 after the MSB there is two consecutive is zeros, but according to the proposed scheme, the pruning is executed only after the LSB is set. Thus, the two zeros are abandoned 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 compactness 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 discarded from the BMVs.

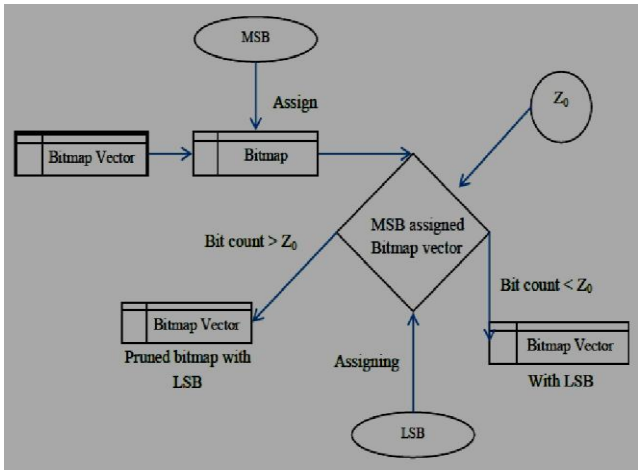


Figure 4: Adaptive pruning

C. Random '1' bit computation

The AP is an early 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 technique 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 level, the proposed method uses a random 1 bit process. In similar to the earlier approach the proposed method also uses a bitwise AND operation between the BMVs. The values of the BMV are also changed by XORing the BMVs with the result of AND operation.

$$R = X \wedge Y; \qquad X = X \wedge R; \qquad Y = Y \wedge R$$

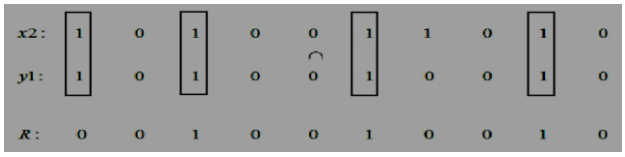
Here R represents the result of the bitwise AND operation and X and Y are the BMVs used for the AND operation. As mentioned in previous sections, the proposed method uses a random 1 bit operation on the BMVs before the AND operation. The BMVs are obtained from the AP technique. Primarily, select the n random 1 bit from both vectors, the n values should be related to the Z_1, the count 1 bit TH. If there are any vectors with 1 bit count less than Z_1, that vector is

discarded. 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 unrelated vectors and such vectors can be pruned from the BM indices. Consider the following e.g.

X □	***Y*** □
***x*1 : [01011]**	***y*1 : [10010010]**
***x*2 : [1010011001]**	***y*2 : [11001]**
***x*3 : [0000000110]**	***y*3 : [000110]**
Priority queue is constructed as,	
X □	***Y*** □
***x*2 :[1010011001]**	***y*1: [10010010]**
***x*1:[01011]**	***y*2 : [11001]**
***x*3 :[0000000110]**	***y*3 : [000110]**
Z_1 = 2;	

X □	***Y*** □
***x*2 :[1010011010]**	***y*1: [0010010010]**
***x*1:[01011]**	***y*2 : [11001]**
***x*3 :[0000000101]**	***y*3 : [000110]**

Here *x3* and y3 are pruned from the BMI, because they do not satisfy the conditional parameters such as Z_1. According to the PQ 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,



If R has 1 bit count greater than the Z_1, then it is added to the IB results. The R is selected for XORing with the attributes X and Y, if any vector possess 1 bit count higher than the TH after XOR operation, that vector added to the BM 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.

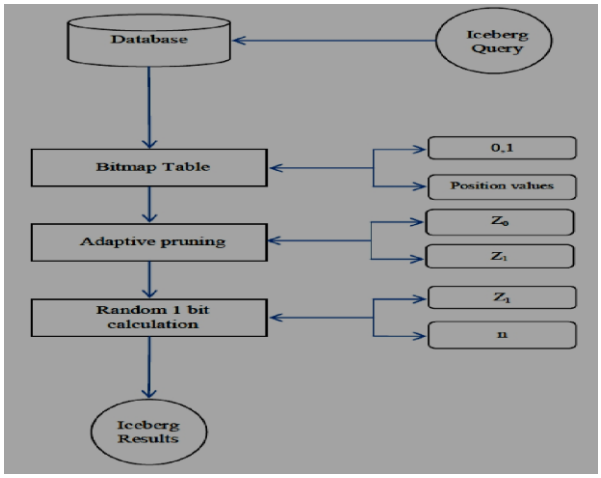


Figure 5: Block diagram for effective IB evaluation

The above Figure 5 shows the architecture of the of the proposed IB 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 meticulous phase. The processing of an IB query to IB result is shown in the architecture.

IV. RESULTS

The proposed method deals with the clustering of data based on the ABC algorithm. The method proposed incorporates the FCM function with ABC algorithm for getting better effectiveness. The performance of the proposed approach is evaluated in the following section under different evaluation criteria.

A. Dataset description

The proposed scheme uses two types of DS for evaluating the IB results. The DSs used are a synthetic data and real data. The DSs are extracted from the Frequent Item set Mining Dataset Repository **[13]**. The DS that is used in the proposed approach is the Retail DS. The retail DS consists of retail market basket data [**2]**. The performance evaluation of the proposed scheme is based on the number of tuples and the TH values. The tuples represent the number elements, which an attribute consist or the number of distinct value possessed by the attributes. The TH value mentioned here is referred to the IB TH. The main evaluation factor that is considered here is the time for execution.

B. Performance evaluation

The performance evaluation of the proposed method, mainly deals with less time for and operations. Thus the time of execution is concentrated more on the advance. The two DS are tested with the proposed IB evaluation algorithm. The responses of the DSs according to the proposed method are detailed in the following graphs. The testing is conducted on different number of tuples in the DS. The responses of the proposed IB evaluation are different for different number of tuples.

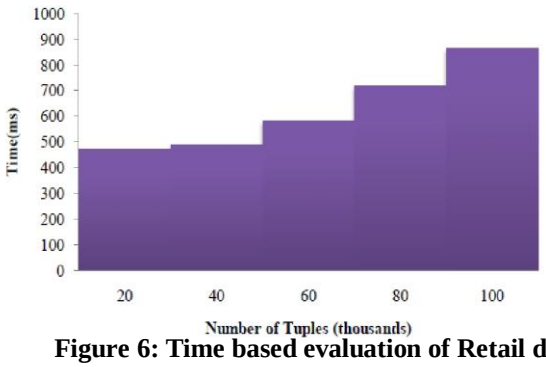


Figure 6: Time based evaluation of Retail dataset

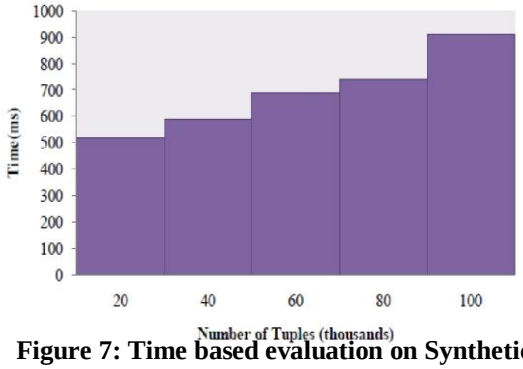


Figure 7: Time based evaluation on Synthetic dataset

The Figure 6 shows the responses of the retail DS to the proposed IB evaluation algorithm. The retail data is evaluated by dividing 5 partitions with uniformly growing 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 equivalently high. The Figure 7 represents response of the synthetic DS according to the proposed IB evaluation algorithm. The response of the synthetic data is similar to the retail DS, 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. In the case retail DS the time required for executing 120000 tuples with the proposed iceberg algorithm is 857 mille seconds.

The testing is extended to appraise the sharpness of the proposed IB query evaluation algorithm. The IB TH has been changed to different values to evaluate the input DSs. The DSs responses are taken according to four different values of the IB TH. The TH value is set according to the count of the attributes since the IB query is subjected for the "COUNT" operation.

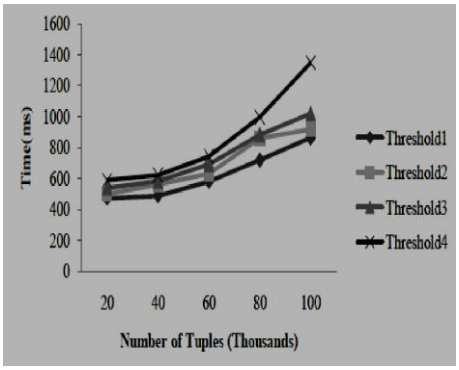


Figure 8: Responses of Retail dataset

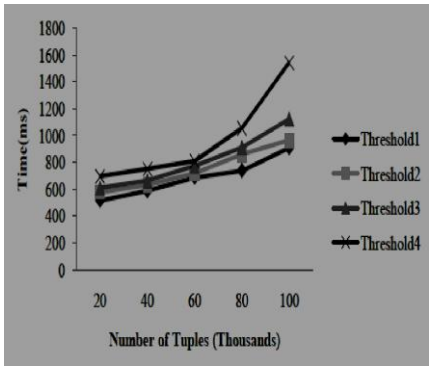


Figure 9: Responses of synthetic dataset

The Figure 8 shows the graph plotted according to the responses from the retail DS for different TH values. The analysis from the graph shows that, as the TH increases the time for execution of the data also increases. This happens because as TH increases, the algorithm searches for most possible AND operations within the limit of the TH. The above plotted result is based on the COUNT operation in the database. The Figure 9 represents the responses of the T10I4D100K DS according to the different TH values. This section includes the comparative study of performance of the proposed IB evaluation algorithm and an existing IB evaluation method. The existing technique refers to the efficient IB query evaluation using compressed BMI. The comparison study is based on the responses of retail DS with two different

algorithms. Both the algorithms are based on the BMI table and the AND operation among the attributes. The plan of the algorithms is to decrease the execution time by reducing the unwanted AND operation.

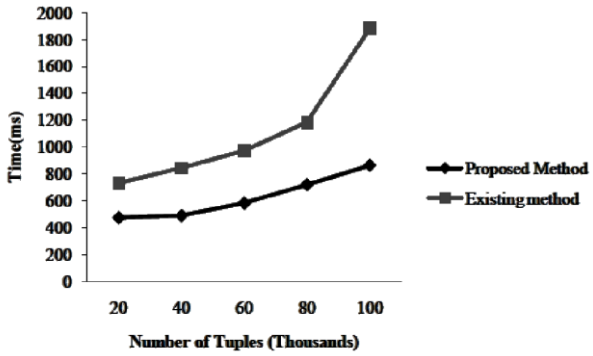


Figure 10: Comparative study

The Figure 10 shows the comparative analysis of the proposed IB evaluation algorithm and the existing IB evaluation algorithm. The analysis shows that time required for execution of 120000 tuples by the proposed approach are 857 milliseconds; on the other hand, the same for the existing method is 1784. In both the cases the time is increasing according to the increase of the number of tuples. Thus it is stated that the proposed IB evaluation has quick response than the existing approach..

V. Conclusions

In the field of IR, the data retrieval from the DB 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 DBs. IB query are similar queries which uses aggregate function and conditional clauses to get the data from the DBs quickly. In the proposed technique, an approach for IB query evaluation is proposed. The proposed IB query evaluation algorithm uses bitmap index table for the IB evaluation. The algorithm has two methods, one for reducing the number of AND operations by reducing the number '0' bits. The second technique is to improve the effectiveness of AND operations by randomly selecting the 1 bit values. The testing is conducted on two DSs in order to evaluate the performance of the proposed IB query evaluation algorithm. In the case of retail DSs, the time required for processing 120000 tuples is 857 milliseconds. The futuristic enhancement can be applied to the proposed technique by processing the PQs using some learning algorithms. In another way, the execution time can also be decreases by eliminating large number of fruitless bitwise-AND operations.

ACKNOWLEDGMENT

The author would like to express their sincere gratitude to the Management of **Vaageswari College of Engineering, Karimnagar** for their constant encouragement and co-operation

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 Early Grouping", 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/

