Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPUTING ICEBERG QUERIES HAVING NON ANTI MONOTONE CONSTRAINS WITH BIT MAP NUMBER

[1]PALLAM RAVI, [2] DR. D. HARITHA , [3] DR. NIRANJAN POLALA

[1] PhD scholar, K L University ,Guntur, Andra Pradesh , India
[2] Computer Science & Engineering, K L University ,Guntur, Andra Pradesh , India
[3] Computer Science & Engineering, KITSWarangal, Telangana, India

E-mail: [1] *satishpallam@gmail.com* , [2] *haritha_donavalli@kluniversity.in* , [3] *pnr.cse@kitsw.ac.in*

Abstract

Computing aggregation values of user interesting attributes are important in decision and knowledge discovering systems .In real world the users are interested only view to the aggregation values which meet some constrain this type of queries are call iceberg queries. We can optimize memory requirement and CPU time for computing iceberg Queries having anti monotone constrains by eliminating non target set, but for Non anti monotone constrains we cannot eliminate non target sets, so it requires huge memory for answering the query, we propose a algorithm which produce target results using minimum memory and CPU time for computation by bit map numbers, explain with a sample iceberg query

Keywords: *Non Anti Monotone, Iceberg Query, Anti Monotone , Bit Map Numbers*

1. INTRODUCTION

The iceberg queries are very important in data mining association rule generation which produce small set of results for example super market analyst find the relationships between products and regions which having total sales above 20 Lakhs from sales data base, this particular query is "SELET product ,region, sum(sales) FROM sales GROUP BY product, region where HAVING SUM(sales)>20 Lakhs" ,the name iceberg query introduced by Fang et al.in[1].The general form of an iceberg query on a relation R($A_1,A_2,A_3,\ldots A_n$) is

SELECT $A_1,A_2,A_3,\ldots A_m$,AGG(*) FROM R
GROUP BY $A_1,A_2,A_3,\ldots A_m$
HAVING AGG(*) >= T

$A_1,A_2,A_3,\ldots A_m$ represents subset of attributes in relation R and referred grouping attributes, AGG represents an aggregation function such as MAX,MIN,SUM ,COUNT AVERAGE and T represents threshold, in this paper , we computing iceberg queries having. Non anti monotone aggregation [2] function such as AVERGE,

Anti monotone property is if a attribute value cannot meet the constraint, its none of its supper sets can meet the constraint, we are explaining with an example on relation R (Table1)

Table 1: Relation R

Tid	*A*	*B*	C
1	A_1	B_1	7
2	A_1	B_1	3
3	A_1	B_2	1
4	A_1	B_3	3
5	A_2	B_3	2
6	A_2	B_2	3
7	A_2	B_2	5
8	A_2	B_1	5
9	A_3	B_2	1
10	A_3	B_1	7
11	A_3	B_1	1
12	A_3	B_3	2

Let us take aggregate attribute is C, anti monotone constraint SUM(*)>8 ,for attribute B_3, $SUM(B_3)=7$ (3+2+2) ,

it will not meet the constrain, none of its super sets $SUM(B_3,A_1)$,=3,$SUM(B_3,A_2)$=2,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SUM(B_3 ,A_3),=2 will not meet the constraint, this is advantage ,there is no need of computing all super sets of B_3 by this we can reduce memory requirement, for Non anti monotone constraint like AVG(*)>4, for attribute A_1,AVG(A_1)= 3.5 [(7+3+1+3)/4] even thought it not meet the constrain its supersets like AVG(A_1,B_1)=5 [(7+3)/2] may meet the constrain

The iceberg query produce small result set due to the constrain, because of the small result set , iceberg queries can potentially be answered quickly even over a very large data set. the relational database systems nowadays (eg DB2,Oracle,SQL Server, etc) are all using general aggregation algorithms [3].[4],[5] to answer iceberg queries by aggregating all tuples and then evaluating the HAVING clause to select the iceberg results .full aggregate results cannot fit in memory for large data set

Iceberg queries algorithm are hybrid , multibuckets ,which are extending the probabilistic techniques proposed in[6].sampling/bucketing method is used to predict valid groups, which possible false positive and false negative .then ,efficient strategies are designed to efficiently correct false positive and false negatives to retrieve the exact result .Bae and Lee [7]design a partitioning algorithm for computing a specific type of iceberg queries computing average of aggregate values .All these techniques are tuple-scan based , Bin He et al[8] proposed compressed bitmap index technique ,but it does not works Non anti monotone constrains ,we are reducing memory requirement and computation time by bit map numbers for computing iceberg queries having Non anti monotone property.

The remaining section of this paper are structured as follows: we discuss bit map number in section 2, section 3 describe Non anti monotone algorithm ,

2. BIT MAP NUMBERS

We introduce new bit map numbers ,this number represent the tuple attribute values unlike bit map index represent only one attribute values in different tuples, this calculate the assign different binary numbers to attribute values of a discrete Attribute(no of bits require represent maximum cardinality of attribute) , concatenate all this binary number with some order, this same order used for all tuples, and calculate number using binary system, which reparsent in algorithm ***Bit Map_ Numbers()***

Example2.1 find bit map Number of Tcid=1 of relation R in Table 1 as follows
Maximum cardinality discrete Attribute (B)=3 ,2 bits required to represent the 3 different values , the order is A, B.
Assign binary numbers to Attribute Values
A_1=01 B_1=01
A_2=10 B_2=10
A_3=11 B_3=11

Tid	***A***	***B***
1	A_1	B_1
	01	01

Bit map Number of Tcid=1 is $(0101)_2 = 5$

All bit map Number of realtion R in table 1 is in Table 2

Table 2: Bit Map Number of Relation R

Tid	***A***	***B***	C	**Bit Map Number**
1	A_1	B_1	7	$(0101)_2 = 5$
2	A_1	B_1	3	$(0101)_2 = 5$
3	A_1	B_2	1	$(0110)_2 =6$
4	A_1	B_3	3	$(0111)_2 =7$
5	A_2	B_3	2	$(1011)_2 =11$
6	A_2	B_2	3	$(1010)_2 =10$
7	A_2	B_2	5	$(1010)_2 =10$
8	A_2	B_1	5	$(1001)_2 =9$
9	A_3	B_2	1	$(1110)_2 =14$
10	A_3	B_1	7	$(1101)_2 =13$
11	A_3	B_1	1	$(1101)_2 =13$
12	A_3	B_3	2	$(1111)_2 =15$

The advantage of this Bit map number are :1) Easily identify the possible combinations of tuple to calculating aggregation functions with the same bit map number 2)Representing database information with this numbers

Assume size of Attributes Tcid,A,B,C in Realtion R are 2bytes ,3bytes ,4 bytes, 1 bytes respectively, the discrete attributes A, B represents bit map numbers require 4*12 =48 bits means 6bytes only instead of 12*(3+4)=84 bytes.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. NON –ANTI MONOTONE ALGORITHM

The algorithm contain three phases
Phase I: calculate Bit Map Numbers for Discrete Attributes **Algorithm1 :*Bit Map_ Numbers()***
Phase II: calculate the Non anti monotone Aggregate
values and eliminate the which are not meet the constrain
Algorithm2: *Clac_aggrigate()*
Phase III: decode Bit Map Numbers into Attribute Values **Algorithm 3: *Decode_ BitMap()***

This algorithm explain which below iceberg query on relation R (Table 1)

SELECT A,B ,AVG(C) FROM R GROUP BY A,B HAVING AVG(C)>=4

Algorithm 1: Bit Map_ Numbers(R,od)
Input :R:raltion ,od:order of attributes
Out put:Bit map Numbers
*1, find **m** for require bits represent higest cordinality of attribute values*
*2, **for** each discrete attribute **do***
*3, assign binary number to each attribute values using **m** bits*
*4, **for** each **T** tuple in R **do***
5, c=concat(all Attribute binary values)
6, $n=(c)_2$
7, stored n along with aggregate attributes: r1

Algorithm 2:Clac_aggrigate(r1)
Input:r1 bitmapnumber with aggregate values
Output:target bit map numbes: rs
*1,**for** same values of r1.bimapnumber:**Tr** **do***
*2, **v**=AGG (**Tr**)*
*3, if **v** meet constrain C*
4, store (tr.bitmapnumber ,AGG())in rs

Algorithm 3:Decode_ BitMap(rs,m,od)
Input:rs result set
Output: decodes the bitmap number
*1,for each **rs.bitmapnumber** do*
*2, group **m** bits(**g**) in **rs.bitmapnumber***
3, decode $(g)_2$ value in od order

3.1 Phase I

Answer the above query using Non –anti monotone algorithm the phase is explained in section 2 , the output of this phase is *r1* is show in *table3*

Table 3

C	Bit Map Number
7	$(0101)_2 = 5$
3	$(0101)_2 = 5$
1	$(0110)_2 =6$
3	$(0111)_2 =7$
2	$(1011)_2 =11$
3	$(1010)_2 =10$
5	$(1010)_2 =10$
5	$(1001)_2 =9$
1	$(1110)_2 =14$
7	$(1101)_2 =13$
1	$(1101)_2 =13$
2	$(1111)_2 =15$

3.2 Phase II

Algorithm clac_aggrigation need only same values of bitmapnumber tuble in *table 3* ,so we compute the aggregate values with minimal memory and increase the computation time also.

Table:4

Iteration	C	Bit Map Number	AVG(c)
1	3	$(0101)_2 = 5$	5
	7	$(0101)_2 = 5$	
2	1	$(0110)_2 =6$	1
3	3	$(0111)_2 =7$	3
4	5	$(1001)_2 =9$	5
5	3	$(1010)_2 =10$	4
	5	$(1010)_2 =10$	
6	2	$(1011)_2 =11$	2
7	1	$(1101)_2 =13$	4
	7	$(1101)_2 =13$	
8	1	$(1110)_2 =14$	1
9	2	$(1111)_2 =15$	2

Clac_aggrigate() compute aggregation in each iteration for the same values of bitmapnumber which shown in *table 4,*the final output of this phase is

Bit Map Number **AVG(C)**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$(0101)_2 = 5$ 5

$(1001)_2 = 9$ 5

3.3 Phase III

This is decoding the bitmap numbers in results set ,it down as follows, group the bits(**m** bits), find its decimal equal and find appropriate attribute values based on encoding Attribute order

In our example query m=2,encoding order id A ,B (left to light)

0101 $01=A_1$, $01\ B_1$

1001 $10=A_2$,$01\ B_1$

The final results of iceberg query are

$A_1\ B_1$ 5

$A_2\ B_1$ 5

4. COMPARISON WITH BIT MAP INDEX

Computing iceberg queries using bit map index[7] 1)the size of Bit map index of a attribute values is depending on number of tuple in data set ,our bit map number is depending on cardinality of attributes only it is very smaller compare to bitmap index, 2)the memory required in bit map index , attributes (at least two) bit map indexes and aggregate attribute values ,in our method only need aggregate attributes and one bit map number 3) multi pull times need to computation of aggregate function but in our algorithm only one time

5. OPTIMIZATION

In Bit Map_ Numbers algorithm can use decimal number for assigning attribute values,a compute Bit map numbers
as $n= A_{1*}(2^m)^{n-1}+ B_{1*}(2^m)^{n-2}+\ldots\ldots$

in over example m=2 ,n=2
$A_1=1$ $B_1=1$
$A_2=2$ $B_2=2$
$A_3=3$ $B_3=3$
$A_1B_2=A_{1*}(2^m)^1+ B_{2*}(2^m)^0$
$= 1_{*}(2^2)^1+ 2_{*}(2^2)^0$
$=1*4 +2*1$
$=6\ (0110)_2$

In *Decode_ BitMap* D is bit map numbers in rest set in our example decoding is as follows

m=2,n=2

$A=D/(2^m)^{n-1}$ $B=D/(2^m)^{n-2}$ .....

6. CONCLUSIONS

This paper present an efficient algorithm saving computation and memory requirement by void multi pull times computing aggregate functions, it need only one time aggregation value computing and direct point the tuple for computing aggregate values
Our algorithm is not sensitive to iceberg query constrain, it give better performance for high cardinality attributes than low cardinality attributes, it works on anti monotone constrains also

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..
[2] R. Agrawal, T. Imielinski, and A.N. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 207-216, 1993
[3] G. Graefe, "Query Evaluation Techniques for Large Databases," ACM Computing Surveys, vol. 25, no. 2, pp. 73-170, 1993.
[4] 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.
[5] P.-A. Larson, "Grouping and Duplicate Elimination: Benefits of Early Aggregation," Technical Report MSR-TR-97-36, Microsoft Research, 1997.

[6] K.-Y. Whang, B.T.V. Zanden, and H.M. Taylor, "A Linear-Time Probabilistic Counting Algorithm for Database Applications," ACM Trans. Database Systems, vol. 15, no. 2, pp. 208-229, 1990.
[7] Bin He, Hui-I Hsiao "Efficient Iceberg Query Evaluation Using Compressed Bitmap Index", ieee transactions on knowledge and data engineering, VOL. 24, NO. 9, september 2012