A Survey: Computing Iceberg Queries

Pallam Ravi*, D. Haritha1, P. Niranjan3
1Scholar KLEF, Guntur
2Professor in KLEF
3Professor in KITS Warangal
*E-mail: Satishpallam@gmail.com

Abstract

Knowledge discovery, tasks are deals with huge no of records. Queries are needed to identify unique attributes values and their aggregate that is above a predefined threshold from this huge number of records. This type of queries is called iceberg queries. Iceberg queries requires huge amount of main memory and takes longer time to answer the query. As computer system has limited amount of main memory, the processing of iceberg queries is a challenging task. This paper discusses different methods that are in literature for processing iceberg queries, we also explore pros and cons of these methods and future scope.

Keywords: Aggregation functions, Bitmap Index, Iceberg Queries, Anti monotone, Bit map Numbers.

1. Introduction

For understand trends in business needs to find insights, for it needs to perform some analytics on business data(historical),it reveals the relations among data features(attributes), this process called as Knowledge discovery process), for identify the relation among attributes need summarized information, which calculated with aggregation functions on one or more attributes, we interest-ed on those have above the user provided threshold value, this type of queries called iceberg queries. Aggregate value above threshold value gives more important information.

For example in college principal wants find relationships between section, course on students marks data, he his interested in which section, course get pass more than 80 students are pass from a section, this will represent as iceberg query as : select section, course, COUNT(*) from marks group by section, course where COUNT(*)>80 and result=”pass”. Here COUNT() is aggregation function, threshold value is 80, the results set include only section, course groups exceeding 80.

The general form of iceberg queries as follow
SELECT A1,A2 … An FROM R
GROUP BY Ai,Aj,… Ak
HAVING AGG() >T

Where R is relation which contain A1,A2 … An attributes and T is user provided threshold

Example 1: the college principal wants which branches have above 50 students admissions from admission database
Select Branch_Name, COUNT() from A
GROUP BY Branch_name
HAVING COUNT()>50

Iceberg queries are widely used many application using such as data warehouse, data mining, multimedia databases and embedded systems

The iceberg query characteristics
1) aggregation function on one or more attributes computation
2) execute on large data set
3) contain large unique attributes combination (domain size)
4) small result set which aggregate values is above threshold.

Problems facing iceberg queries while executing are 1) need to execute within limited memory means domain size is greater than memory 2) Aggregation values computation takes huge time. The global research objective of iceberg queries execution is reduce time takes to execute query within limited memory only

Rest of paper are structure as follows, in section 2 discussed related work done on iceberg queries, different authors contribution with respective aggregation functions, data scan and computing environment discussed in section 3, in section 4 discussed about different method with respective aggregation functions, data scan and computing environment, section 5 gives future scope in iceberg queries.

2. Related Work

The first study about iceberg queries by fang et al[1] in it coarse count and sampling methods are used to answer the iceberg queries, but it suffers false negatives and hybrid multi bucket algorithm proposed in [1], these are extended with probability techniques whag et al [2].

But [1][2] are not work for average queries, Bae and Lee[3] are proposed partition algorithms (BOP and POP) for average iceberg queries computation, disadvantage of these algorithm is many data scans are required, leela et al[4] comparative study using sort merge aggregate, ORACLE and hybrid hash aggregate methods, which reveals sort merge aggregate gives better performance.

Ferro et al.[5] use bitmap index which suffer massive empty Bitwise AND operation , to reduced mass empty bitwise And operations. Ferro et al.[6] proposed dynamic pruning algorithms, to reduce number of Bitwise and operations, Shanker et al proposed different algorithms [7][8] differencing push and pop operations, cache based [9] check point mechanism which reduce number of Bitwise AND operations.

Above development on iceberg query computation goal in direction of usage of aggregation functions properties, type data base scan and computing environment to reach global goal of iceberg queries.

3. Author Directions

This is section explore different authors contribution, we classified their contribution as aggregation function properties, type of data scan and computing environment.

3.1 Aggregation functions properties:

Aggregation function like AVERAGE, SUM, MAX, MIN and COUNT... we these are divided into anti-monotone and non anti-monotone, anti-monotone Aggregation functions use Apriori[14] property, on anti-monotone are not able to use Apriori property. Example of anti-monotone aggregation functions are SUM, MAX, MIN and COUNT for non anti-monotone aggregation functions are AVERAGE, STDIV.

Takes advantage of Apriori property for Computing anti-monotone iceberg queries (iceberg query with anti-monotone Aggregation functions) by this the pruning of computing Aggregation functions will take place due this reduce time for generate query result set.

Non anti monotone aggregation iceberg queries are not takes advantages of threshold on AVERAGE values as SUM, MAX, MIN and COUNT Aggregation function (anti monotone aggregation functions) average iceberg queries need to compute AVERAGE for all unique grouped attributes then apply threshold constraint on these AVERAGE values maintain a counter bucket for each unique grouped attributes. Normally required a counter bucket are not maintain in memory (characteristic of iceberg query).

[2] proposed a partitioning two methods (BOP & POP). these methods sequentially partitioned data, number of unique values in partition data are less the maximum number counter bucket are handled in memory, each counter bucket have two tuples <value,1>. produce results set by apply threshold constraint on a counter bucket with calculate AVERAGE by use its sum and count values (average=Sum/count), its have two disadvantages those are:

1) two times need to compute AVERAGE value per one candidate unique group attributes(one for selecting candidate unique group attributes, other is to decide actual value of candidate meets threshold constraint)

2) many scan on data(equal to number of partitions)

3.2 Type of Data Base Scan:

for data scan, use tuple based and column based scanning data, in tuple based iceberg query computing is support for small data sets not for large data sets, in column based it works for large datasets due to use bitmap indexing(BI), BI takes lesser memory to represent data, advantage with BI is quick indexed the record values

3.3 Computing environment:

The work done so for single processor except shanker et al[15] is focus on distributed environment, using data shipping and query shipping proposed different algorithms.

<table>
<thead>
<tr>
<th>Shno</th>
<th>Author</th>
<th>Method</th>
<th>Aggregation function</th>
<th>Type of scan</th>
<th>Environment</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tang et al</td>
<td>Coarse-count</td>
<td>Anti-monotone</td>
<td>Tuple</td>
<td>Single</td>
<td>False negatives</td>
</tr>
<tr>
<td>2</td>
<td>Whag et al</td>
<td>Probability</td>
<td>Anti-monotone</td>
<td>Tuple</td>
<td>Single</td>
<td>False negatives</td>
</tr>
<tr>
<td>3</td>
<td>Bae and Lee</td>
<td>Partition</td>
<td>Non Anti-monotone</td>
<td>Tuple</td>
<td>Single</td>
<td>Small data</td>
</tr>
<tr>
<td>4</td>
<td>Leela et</td>
<td>Sort merge</td>
<td>Anti-monotone</td>
<td>Tuple</td>
<td>Single</td>
<td>Small data</td>
</tr>
<tr>
<td>5</td>
<td>Ferro et</td>
<td>Dynamic</td>
<td>Anti-monotone</td>
<td>Column</td>
<td>Single</td>
<td>Bitwise AND operations</td>
</tr>
<tr>
<td>6</td>
<td>Shanker et</td>
<td>Push and pop</td>
<td>Anti-monotone</td>
<td>Column</td>
<td>Single</td>
<td>Bitwise AND operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2,cache based</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,check point</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mechanism</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data shipping</td>
<td>Anti-monotone</td>
<td>Column</td>
<td>Distributed</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Methods Comparison
4. Methods Comparisons

We are presenting different method’s Advantages and disadvantages, which helps to future scope. Representing method in table 1, which explain author’s direction .

5. Conclusion

The paper explored the various methods for processing iceberg queries. This paper mainly focused the iceberg query processing with respect to aggregation function, data scan and computing environment. There is lot of research needed to focus on handling large value of data for non anti-monotone aggregation iceberg queries on single and distributed computing environments.

References