Efficient Iceberg Query Evaluation for Structured Data using Bitmap Indices

Ms. Archana G. Narawade \(^a\), Mrs. Vaishali Kolhe \(^b\)
\(^a\) PG student, D.Y.Patil college of Engineering, Akurdi, Pune
E-mail: archanasaid@rediffmail.com
\(^b\) PG Guide D.Y.Patil college of Engineering Akurdi, Pune

Abstract—A data warehouse is a relational database that is designed for query and analysis over big data. Mostly it contains historical data derived from transaction or OLTP data. It separates analysis workload data from transaction workload data and enables an organization to consolidate data from several sources. The data warehousing concept was intended to provide an architectural model for the flow of data from operational systems to decision support environments like Insurance fraud analysis, latest market trend, profit analysis etc. Data which is retrieved from data warehouse is used by analysts to make business related decisions. To retrieve any data from a big database requires at least one scan over entire database/data warehouse. To scan the big data it should be brought into main memory which quite infeasible as main memory is not too big to accommodate millions or billions of data. Working with aggregate functions requires full table scan which is very inefficient method. Bitmap index can be built on attributes is nothing but stream of 0’s and 1’s which indicates presence(1) or absence of data (0). Aggregate queries gives faster result on which bitmap index is built by rearranging the columns in efficient order.

Index Terms—Online Transaction Processing, Bitmap index, dynamic pruning, data analysis, big data, aggregate queries.

I. INTRODUCTION

Business analysis is a research discipline of identifying business needs and determining profit value out of it. Often to find out market trend business analysts go through data stored in data warehouse by retrieving information by using aggregate functions. Group by clause in aggregate functions involves multiple columns from distinct relational table by natural join\(^{(10)}\).

If bitmap index is built on attributes then join will give faster results\(^{(3)}\). This paper presents efficient arrangement attributes so that Aggregate query result will be retrieved in minimum amount of time.

II. LITERATURE SURVEY

Different approaches have been proposed by researchers to address querying data from data warehouse but time to retrieve data is still a crucial factor. Bitmap indices have gained wide acceptance in data ware-house applications and are an efficient access method for querying large amounts of read-only data\(^{(4)}\). Over the last three decades many different index data structures were proposed to optimize the access to database or data warehouse. Some of these are optimized for one-dimensional queries such as the B+tree whereas others are

© Elsevier, 2013
optimized for multi-dimensional queries such as the bitmap indices. Recently, bitmap indices have gained wide acceptance in data warehouse applications and are an efficient access method for evaluating complex queries against read-only data. The main approach in bitmap index research focuses on typical business applications based on discrete attribute values.

A data warehouse is a relational database that is designed for querying data and analysis rather than for transaction processing which involves number of data manipulations. It contains historical data derived from transaction data, but it can include data from other distinct sources. It separates analysis workload data from transaction workload data and enables an organization to consolidate data from several sources. Data Retrieval and Analysis is how businesses obtain insight into their operations before making critical decisions. Businesses capture information across multiple dimensions of activity (e.g. sales, inventory, market research) over time. Capturing more information from more sources over time leads to exponential data volume growth. Data retrieval is the process of capturing these ever increasing amounts of information, transforming and formatting the data for its business needs, and loading it into the target data warehouse. This data retrieval process is often referred to as Extract, Transform, and Load (ETL). Efficient and effective data retrieval over the data warehouse can be done efficiently using bitmap indexing techniques\[2\][7].

A. Indexing for data warehouse:

Performance is a critical issue in database and data warehouse environments. Like indexes in books, indexes in databases and data warehouses can provide fast data retrievals of inquiries. B-tree indexing is currently popular in relational-based data warehouses. However, many new indexing techniques are being implemented for the special environments of data warehouses\[5\].

The concept of bitmap index was first introduced by Professor Israel Spiegler and Rafi Maayan in their seminal research "Storage and Retrieval Considerations of Binary Data Bases", published on 1985 (Information Processing and Management, Volume 21, Issue 3, 1985, Pages 233-254). The first commercial database product to implement a bitmap index is Computer Corporation of America's Model 204. Patrick O'Neil implemented the bitmap index around 1987. This implementation is a hybrid between the basic bitmap index (without compression) and the list of Row Identifiers (RID-list). Overall, the index is organized as a B+tree. When the column cardinality i.e number of distinct records is low, each leaf node of the B-tree would contain long list of Row IDs. In this situation, this database index requires less space to represent the Row ID-lists as bitmaps. Since each bitmap represents single distinct value. Bitmap indices are mainly used in evaluating Iceberg queries\[6\].

An iceberg query performs an aggregate function over an attribute (or set of attributes) and then eliminates aggregate values that are below some specified threshold.

\[
\text{SELECT } C_1, C_2, \ldots, \text{AVG}(C_4) \\
\text{FROM } R \\
\text{GROUP BY } C_1, C_2 \\
\text{HAVING } \text{AVG}(C_4) >= TH
\]

Where \(C_1, C_2, \ldots\) are distinct columns from relation \(R\) \(\text{AVG()}\) is aggregate function which is going to calculate average value on the column \(C_4\).

B. Use of bitmap index to answer iceberg queries:

1. Bitmap indices can avoid massive disk access on tuples. Using bitmap indices, there is only need to access bitmap indices of the aggregate attributes (i.e. the attributes in the \text{GROUP BY clause}).
2. Bitmap indices operate on bits rather than actual row values. Bitwise operations are very fast to execute and can often be accelerated by hardware\[8\].
3. Bitmap indices have the advantage of leveraging the anti-monotone property of iceberg queries to enable index pruning strategies. Iceberg queries have an important anti-monotone property for many of the aggregation functions. For example, if the count of a group is below Threshold value, the count of any super-group of the aggregate function must be below Threshold value. Each iceberg result can be produced by doing a bitwise-AND between the bitmap vectors representing each value in a group and counting the number of 1 bit in the resulting bitmap vector\[2][3]\.

C. Related work

Approaches to Evaluate Iceberg queries
1. General aggregation algorithm: First aggregating all tuples and then evaluating the HAVING clause to select the iceberg result.
2. Multi-Pass aggregation algorithm: Used when full aggregate result is not able to fit into main memory \(^4\).
3. Tuple-Scan based approach: This approach requires at least one table scan to read data from secondary memory. The focus on reducing the number of passes when the data size is too large.

None of the above has effectively leveraged the property of iceberg queries for efficient and accurate query processing. The tuple-scan based scheme often takes a long time to answer aggregate or iceberg queries, especially when the database is too large. It gives massive empty bitwise-AND results problem.

This paper is to aim at answering iceberg query efficiently using bitmap indices. Specifically, focus is on developing an index-pruning based approach to compute iceberg queries using bitmap indices. Bitmap indices provide a vertical organization of a column using bitmap vectors containing 0’s or 1’s. Each vector represents the occurrences of a unique value in the column across all rows in the database or table.

III. PROPOSED SYSTEM AND IMPLEMENTATION

![Figure 1 System Design](image)

Figure 1 shows the system architecture diagram for the proposed system. Proposed system divides into following module.

Module I: Data Extraction Phase
Module II: Pruning Phase

1. Data Extraction: Huge database will be the input to the system in the form of SQL query with aggregate functions like MIN(), AVG() etc.
2. Dynamic Pruning: Anti-monotone property is used to reduce empty bitwise AND operations.
3. Find 1-Bit Position: Extension of dynamic pruning to find aligned vectors
4. Find Aligned Vectors: Priority queue will be formed to find aligned vectors
5. Evaluation

In this algorithm a Greedy algorithmic Approach is used in which the algorithm follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. Algorithm 1 contains the steps for obtaining column combination for which minimum time is required.

**Algorithm Name:** Select combination with minimum time

**Input:** Iceberg Query with more than 2 columns in SELECT statement

**Output:** system will produce different combinations for which data retrieval time is minimum.

**Steps:**
1. Initialize variable count
2. Submit the query
   - For each value from 1 to n
     - For each combination calculate the time required to execute query.
3. Display column combination with minimum time
As shown in figure 1 attributes are arranged in such way that data through aggregate function will be retrieved in minimum time.
SQL query can executed for different threshold values for checking efficiency of optimal ordering of attributes.

IV. EXPERIMENTAL SETUP

The experiments are conducted on a machine with a Pentium 4 single core processor of 3.6GHz, 2.0 GB main memory and 7200rpm IDE hard drive, running on Windows OS platform. All algorithms are implemented in C sharp and VB using .net 2010. Oracle is used as database for string relational data. Algorithms are evaluated on some standard dataset from World Bank website. Experimental results are calculated for different columns and various threshold values for checking the efficiency of reordering of attributes in aggregate query on SPORTS dataset.

V. RESULTS

Query retrieval time is evaluated on following query
Example:

```
Select team, position, player, avg(salary)
From SPORT
Group by team, position, player
Having avg(salary) > TH
```

As given in above query such a order of attributes is found out so that in minimum time data can be retrieved for given threshold value (TH).

For example.

If team, position, player requires 15 msec
position, player, team, requires 20 msec etc.

Such combinations are found out and the columns combination for which minimum time is required is finalized.

For finding exact combination query is executed 20 times (User Defined can be changed for better accuracy if no. of records are in huge quantity).

After finding out exact combination of columns with bitmap indices same combination query is fired without bitmap which gives data retrieval time more than bitmap indices. These results are plotted as shown in figure 2

Figure 2 shows minimum time required for the attribute combination for above mentioned SQL query on X-Y axis.

The attributes combination is going to give data retrieval in minimum time as long as requirement of data is not modified.

If requirement for information is changed then we need to find out attributes combination for the respective requirement.

![Figure 2 Query time analysis with and without bitmap index on Query attribute](image-url)
Figure 2 shows time required to execute SQL query on multiple attribute with and with out bitmap index on attribute. Time required to execute same SQL query with bitmap index requires less time as compared to query without bitmap index.

VI. CONCLUSION

Proposed system efficiently evaluates SQL aggregate query by properly rearranging attributes in the SELECT statement for aggregate functions: COUNT(), AVG(). After rearrangement of the attributes query retrieves the result in minimum time. Retrieved result carries valuable information. In today’s world taking decision to run a business is crucial factor. Business analyst always required to analyze the historical data so that they can take right decision so as to get best results. SQL queries with aggregate function always returns small data values out of large dataset which carries important results. The proposed system is going to help business analyst make proper decision as Iceberg Query is giving result in minimum time. Proposed system also reduces I/O operations as bitmap indexes are used on the columns for which data to be retrieved.

VII. FUTURE WORK

Implemented system work can be extended for retrieving data using aggregate functions with more than three attributes on which bitmap compression technique can be implemented so that memory can be efficiently utilized.

REFERENCES