Efficient Processing of Queries using Filtered Bitmap Index with Multi-Join Multiple Set Predicates

A. Regita Thangam, S.John Peter

Abstract*: *Efficient query process is an essential task in numerous environments that relate large sum of information. The performance degradation occurs when the sum of information is increased. It will further degrade when the amount of joins in the queries is increased. These problems emphasize a need for good query processing approach. Thus, in this report, we take a various method to optimize the multi-join query with multiple set predicates in Data warehousing environment. So we have proposed an effective algorithm as Filtered Bitmap Index with multi-join multiple set predicates processing approach and examine the time complexity on huge data set with multiple tables. In this approach, the multi-join query is processed by selecting the tabular array based on their level number from lower to higher. A simple rewritten query was created from the given complex query exploitation uses the lowest level table and executed. If the result exists then only continue the join processing in the rewritten query, by taking the next lower level table from the complex query and do the execution. The ratio of our technique is to demonstrated with moving experiment using WorldCup98 and TPC-H benchmark datasets.

Keywords: Data warehousing, time complexity, multiple set predicates, multi-join query, query processing, optimizing, level number. .

I. INTRODUCTION

Query improvement is the primary work for the execution of compound SQL statement to modify the execution of a relational database. It is also a key technology for many real time applications. information from various business applications are integrated and stored in data warehouse. The Data warehousing spotlight the gaining control of data from different point for access and analysis. In present scenario data storehouse is one of the common basis for the integration and analysis of information in modern enterprises. So it is essential for doing efficient share to the region of query optimization [13].

Query Processing means the whole activity which refer query version into down level instructions. Query optimization way to clutch on the origin, expending idea or evaluation of query, and descent aggregation from the database. In modern days, state of querying the data with the humanities of set-level comparison is high in accumulation storehouse and OLAP applications.

If the set level comparing are performed using presently accessible SQL structure, the ordered query may be more and more complex. Such colonial query are not only hard for the user to explicate but also outcome is too much expensive for valuation [7].

To find some selected records from large data sets is the common job in the collection storage applications. Efficiently respondent these questioning is quite often challenging owed to analyzable the nature of some the data and the inquiring.

The SQL answer is to find the candidate with ability “XML” and “Python”, as follows:

```
SELECT c_candidateid FROM Candidates, Skillset
WHERE c_candidateid = s_candidateid
GROUP BY c_ candidateid HAVING SET(skills)
CONTAIN {’XML’, ‘Python’}
```

From the section query, a dynamical set of content on the concept skills is formed for each specific c_ candidateid, and the groups whose similar SET(skills) contain both “XML” and “Python” are returned as query answers.

The SQL query to discovery the Hindi articles with the authors Mary and Raja only. For this query, the EQUAL operator are used as below:

```
SELECT a_id, a_articlename
FROM Articles, Language, Author
WHERE a_id=l_articleid
and l_id = au_languageid and l_language=’Hindi’
GROUP BY a_id, a_articlename
HAVING SET(au_author) EQUAL {’Mary’, ‘Raja’}
```

We have tables named as Ratings, College, and section. View the following decision making query. It finds the departments whose portion time mean evaluation in 2018 have always been poor (assuming the rating is from 1 to 5) for the colleges STC and SXC:

```
SELECT d_clgname, d_departid
FROM College, department, Ratings
WHERE c_clgid = d_clgid and
d_departid = r_departid and
r_year = 2018
GROUP BY d_clgname,
```

Revised Manuscript Received on July 22, 2019.

A.Regita Thangam, Research Scholar, Department of Computer Science, Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli, Tamil Nadu, India

S.John Peter, Associate Professor & Head, Department of Computer Science, St.Xavier’s College, Palayamkottai, Tamil Nadu, India

d_departid HAVING SET(d_clgname) CONTAINED BY {'STC','SXC'} And SET(r_avgrating) CONTAINED BY {1, 2}

In this above query multiple tables are joined and many number of set assert are used to make the output. In this multi-join query, CONTAINED BY is used to seizure the set-level situation.

Without the scheme of set relate, the query linguistics may be capture by using sub-queries adjoining by SQL set dealing (UNION, INTERSECT, EXCEPT), in timing with join and GROUP BY. Such queries may be rather colonial for users to develop which results so much expensive for assessment. On the adverse, the set predicate construct accordant to personification of the current innovation explicitly enables set-level comparisons. The concise syntax kind query style is simple and also assist the efficient native help of such enquiry in a query engine.

The query syntax also allows comparing the sets defined on multiple attributes. A query with duple set predicates can be based on with the Boolean Operators such as AND, OR and NOT and the aggregated functions such as AVG, SUM and COUNT.

This material is arranged as follows. Subdivision I contains the start of multi-join query with multiple set predicates process, the affiliated activity parturition the stage for our method is discussed in section II. The projected formula for Filtered Bitmap Index with multi-join multiple set predicates processing approach is explained in section III. The research event and relation of the planned rule with the progressive algorithmic program is described in section IV. Finally, section V resolve this report and spotlight some rising directions.

II. Related Work

Query Optimization reference to the work of finding the best execution scheme for a granted query from the set of alternative. There is a broad status for querying data with set predicates.

form set level comparing without any restriction inception by database schema for set predicates. In many applications, there is a need to integrate data and operations that are external to the database. Access to such outside data is supply by a set of interface routines.

To increase the throughput of multiple queries, they can be concurrently executed [4]. The importance of Multi-Query Estimation in the circumstance of relational database query process is explained by J.Chen et al. [1].

An efficient algorithmic rule Filtered bitmap index based method for processing queries with set predicates was planned in [18]. This algorithm has the welfare of saving disk access and the computing time was reduced by reducing the number of iterations.

Swathi Kurunji et al. [11] presented an algorithm for processing the multi-join query and reduce communication overhead in cloud environment. An extension of traditional query rewrite techniques was proposed by Albrecht et al. [2]. They presented the derivability conditions for a large number of relevant cases.

Two or more relations can be concerted with Rank join operators and produce output with the highest aggregate grade. The rank joint difficulty [5] has been dealt in the written material by extending rank aggregation algorithms [3] to the cause of join in the scope of relational databases. The Cost-Aware with Random and Sorted access scheme was proposed by Davide Martinenghi et al. [12] for retrieving the k-combinations with the highest aggregate score that can be formed by joining the results of heterogenous search engines. They optimized the strategy with an additive cost model that see both grouped to access and random access.

Distributed query optimization is one of the main problems in the database area [6]. There is a possibility of more than one equivalent query for a given query. Some of these equivalent queries are better than the others. The expected performance specifies the quality of an algebraic query.

Efficient determination make a critical in a planetary environment where concern ability to the group that are becoming an necessary part of virtually every system. The core of such systems is a data storage warehouse, which stores arts and consolidated data from the transactional databases, supporting complicated queries that reveal exciting information [4].

Javier Tuya et al. [8] proposed a coverage criterion to assess the data in a test database in relation to the SQL query that is executed. The criterion defines a number of test point requirements for the query under test that lead to executable coverage rules obtained by applying a set of transformations to the query. The coverage criterion considers the SELECT clause elements such as selection, joining, grouping, aggregate functions, subqueries and case expressions. It also presented the coverage rules for each SQL operator and their combinations.

Modern-day information systems use a query optimizer to identify the most expeditious plan to execute declarative SQL queries. The function of query optimize rs is critical for the decision-support queries conspicuous in data warehousing and data mining applications. Pawan Meena et al. [9] proposed an abstract of the architecture of a query optimizer and the technical constraints of advanced issues in query optimization.

A global index based optimization strategy for scope query and reasoning was proposed by Hui Zhao et al. [10] and they do some tests to measure the rightness and efficiency at the end. The strategy was first checking whether user requests can be optimized by using the global index knowledge.

Aggregative function technique and Bitmap index based technique was proposed by Chengkai Li et al. [15] to process query with set predicates. . Second technique is more prompt because it focuses on only those tuples which satisfies query premiss and bitmaps of appropriate columns. Such scale structure is applicable on many various types of attributes. This technique

processes queries such as selections, joins, multi-attribute grouping etc.

Jayant Rajurkar et al. [16] formulated a bitmap pruning strategy by using Word Aligned Hybrid (WAH) compression for processing queries which eliminates the necessity of scanning and processing the entire data set. This technique is used for query optimization with set predicates. The set term have various advantages than the set-valued attributes together with set system joins which can support set-level relation.

The survey on most of the query optimization techniques were proposed in the paper [17]. They reviewed most of the recent works carried out in this area. The query optimization is a huge research area and can be prolonged in several ways.

An efficient algorithm Filtered bitmap index based approach for processing queries with set predicates was proposed in [18]. This algorithm has the benefits of saving disk access and the computing time was ablated by reduction the number of restate. In this algorithm the groups and the corresponding sets are formed accordant to the query needs which results in speeds up the query processing.

A.R. Thangam et al. [19] formed prompt algorithm Reduced Function-Based attack for process queries with set relate. Today many query processing experts agree that standard query optimization has reached its limits and needs to be re-thought that many intelligent query optimization techniques must be alter.

An extensive analysis about queries with set predicates for supporting group-level comparisons was presented by A.R. Thangam et al. [20]. The four evaluation methods to process query with set predicates were compared to get the effective approach. It shows that Filtered Bitmap Index Based approach is more time saving to process query with set predicates.

III. PROCESSING MULTIPLE SET PREDICATES

A. Data Warehouse Model

A data warehouse is a cardinal depository of information that can be analysed for making better decisions. Data can be entered into a data warehouse from relational databases, transactional systems, and other sources. Data warehousing contains the process of constructing and using a data warehouse. Star and Snowflake Schema are correspond to the commonly used Data Storage. A Snowflake Schema is an delay of a Major Schema, and it adds additional dimensions in data warehouse model. It is called snowflake because its plot match a Snowflake. The conception tables are normalized which splits data into additional tables.

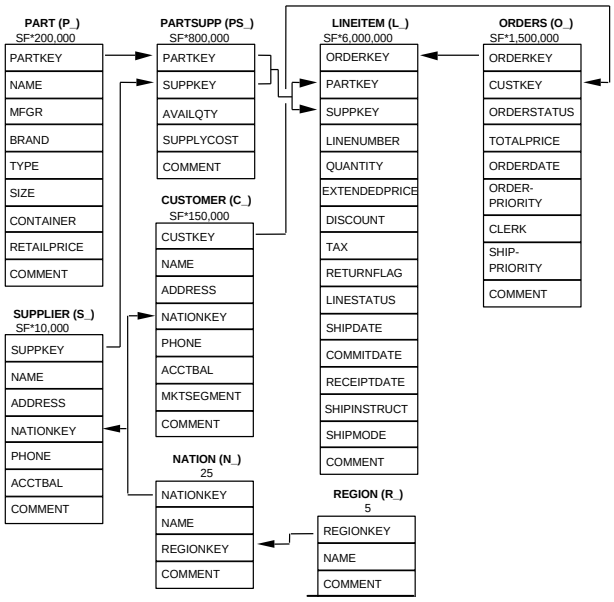

Fig.1 TPC-H Schema (source: [21], page: 13)

In Fig. 1, the prefix of the column names for that table is specified with in the parentheses following each table name. The arrows specify the relation between tables. The number which is specific below each table name represents the number of rows present in that table.

B. Multiple Set Predicates

The SQL syntax is drawn-out to support set predicates [15]. Set predicates compare a grouping of tuples to a fixed values. So it can be fitted into GROUP BY and HAVING clauses. Generally HAVING clause there is a Boolean aspect over multiple orderly aggregated predicates and set predicates related by logic function AND, OR and NOT.

The syntax of multiple set predicates is
SET(v_{11}, . . . , v_{1m})
CONTAIN | CONTAINED BY | EQUAL
{(v_{11}^1, . . . , v_{1m}^1), …. , (v_{11}^n, . . . , v_{1m}^n) }
AND / OR / NOT
SET(v_{21}, . . . , v_{2m})
CONTAIN | CONTAINED BY | EQUAL
{(v_{21}^1, . . . , v_{2m}^1), …. , (v_{21}^n, . . . , v_{2m}^n) },
where $v_{ki}^j \in Dom(v_{ki})$, i.e., each v_{ki}^j is a literal value in the domain of attribute v_{ki}.

The set predicates allow sets to be dynamically formed through GROUP BY and supports CONTAIN, CONTAINED BY and EQUAL.

The SQL query to get the available quantity of parts for each supplier which is manufactured by only MFGR#1-#3 and having the brands, brand #13 and brand #42, but nothing else.

```
SELECT PS_SUPPKEY, SUM(PS_AVAILQTY)
FROM PARTSUPP, PART
WHERE PS_PARTKEY=P_PARTKEY
GROUP BY PS_SUPPKEY HAVING
SET(P_MFGR)
CONTAINED BY
{'MFGR#1', 'MFGR#2',
```

'MFGR#3'} And
SET(P_BRAND) CONTAINED BY {'Brand#13', 'Brand#42'}

It uses two set predicates to handle the given query condition. The result will get the available quantity of parts for each supplier which is manufactured by MFGR#1 - #3 only and having the brands, brand #13 and brand #42 only.

In more number of cases, linguistics of group-level compare may be expressed by using the actual available given SQL syntax without suggested expansion. But the queries resultant will be increasingly complex than need . The complicated queries can be difficult to create using the currently available syntax for users. More important, such complicated queries may be tough for optimizing too. This leads to unnecessary expensive evaluation [11]. The query plans resulting from these may contain multiple inner queries. The recommended syntax with set predicates empowers direct stating of group-level comparisons within SQL, and this makes the easiest formation of query. It also gives effective support to such queries.

C. Filtered Bitmap Index with multi-join multiple set predicates Approach

Our approach is proposed for processing multi-join queries with multiple set predicates. In this material we use the Snowflake representation as in Fig. 1 from "TPC BENCHMARK H Modular Description Alteration 2.17.3". TPC-H is a selection help benchmark which is designed to support the practicality of concern reasoning applications. The representation of an manufacture which can bring off, sell and deal its products worldwide is specified in Fig. 1. This composition focuses on relational data model and subject.

Level_Info table: One of the key relational database query operations is the join operation. It assist the retrieval of information from two or more various tables.

Table 2. Level_Info

TableName	levelno
Region	1
Nation	2
Supplier	3
Part	1
Part_Supp	4
Customer	3
Order	4
LineItem	5

The Table 2 contains the level number of the TPC-H tables specified in Fig. 1. The levelno gives the depth level of the required table. The table with no reference having the levelno as 1. The figure of other tables joined to it starting from the root array, produces the level no of any table. This data is used to choose the tables for join operation based on their level number to execute the query.

The sketch of the algorithm 1 is as below. It is used to evaluate the multi-join queries with multiple set predicates containing the three kinds of set operators {⊇, ⊆, =}. Here LT represents the level info table which contains the information about the level number of each table in the database. The number of tables engaged in the query is specified as n and used as input to this algorithm. The query to be executed may contain multiple set predicates. CV_{11}, ……CV_{1K} specifies the condition values of predicate column V_1 and CV_{21},……CV_{2K} specifies the condition values of predicate column V_2. Likewise the query can have any number of set predicates. And g specifies the group by columns and ⊕a specifies the aggregate columns.

In the first step, it sorts the Levelinfo table LT with the level number in ascending order. Then choose the tables specified in the Query Q based on their sorted level number and put it in an array SLT. Initialize the table Resultset as empty. Then select each table from the sorted array SLT (line 5) and check whether it is a first table in the array SLT (line 6). If it is the first table then executes the subquery related to the current table t_i from the given query. Store the resultant records in a table Resultset. Otherwise, join the subquery related with table t_i with the existing Resultset. Now the subquery contains all the fields and predicates corresponding to the table t_i joined with the existing resultant table Resultset. Execute this query and store the resultant records in a table Resultset. If the new Resultset is empty then no need to continue further processing and exit from the algorithm (line 7). It reduces the consuming time and improves the performance. Otherwise it produces the qualified groups 'g' and their aggregate values '⊕a' as output.

Thus it performs the join operation one by one as per the levelnumber in level_info table. Each time, after performing the join operation, the number of rows returned is checked. If the count is zero, then further processing of the query will be skipped and no qualified group was found. Thus the time complexity will be reduced. It skips the further steps. After performing all the join operations and if it contains the output records, then it will produce the output as in line number 9 of our algorithm. Eliminating the unwanted query conditions from processing, improves the performance of our algorithm. It supports the set predicate operators CONTAIN, CONTAINED BY and EQUAL. The code for algorithm1 - Filtered Bitmap Index with multi-join multiple set predicates processing Approach is as below:

Algorithm 1: Filtered Bitmap Index with multi-join multiple set predicates processing approach

Input:

LT - LevelInfo table,
Query Q = ϒg,⊕a V_1 op {CV_1,……CV_K} And
V_2 op {CV_1,……CV_K}

Output:

Well-qualified groups g and their aggregative values ⊕a

***Steps*:**

1: Sort the Levelinfo table LT with the levelnumber in ascending order
2: Find the number of tables n specified in the given query

3: Choose the tables specified in the Query Q based on their sorted levelnumber and put it in an array SLT.
4: Resultset = empty
5: For each table $t_i \in$ SLT do, i ← 1 to n
6: If i = 1 then
Resultset = Execute the subquery related to the table t_i from the given query
Else
Join the subquery related with table t_i with the existing Resultset
Resultset = Execute the above query
End if
7: If Resultset = empty then
No qualified group found
Exit
End if
8: End for
9: Output qualified groups and their aggregate values
10: End Algorithm

The multi-join process in our approach is explained below. The multi-join complex query is processed by selecting the tables based on their level number from lower to higher. The algorithm first selects the required fields of lowest level table in the complex query and rewrites a simple query using this lowest level table and

its related fields and conditions. Then this rewritten query was executed. Check the total number of records in the output. If it is zero, then need not go to further processing, exit from the loop. Otherwise, if the result exists then only continue the join processing in the rewritten query, by taking the next lower level table from the complex query and do the execution.

IV. RESULTS AND DISCUSSION

The experiments are performed on the Intel I3 processor with 4GB RAM memory. The proposed algorithm uses Multi-join Filtered bitmap index based technique.

A. Experiments over TPC-H Data:

TPC-H benchmark provides a utility dbgen for generating data with various sizes for doing the experiments. So we generated data using this data generator. The efficiency of this algorithm is proved by using TPC-H benchmark dataset. We generated 1 GB of data for the experiments. The Filtered Bitmap Index with multi-join multiple set predicates algorithm is implemented using Matlab.

***Queries*:** We designed two types of queries on this dataset as follows:

Query Type: 1

Here we consider the queries having different number of joins.

***Example 1.1*:**(query with one join)

To find average account balance of the suppliers who belong only to the Nations India and France.

```
Select S_Name, Avg(S_Acctbal)
From Nation, Supplier
Where N_Nationkey = S_Nationkey
Group By S_Name Having
Set(N_Name) Contained By {India, France}
```

This query has a join between two tables - Nation and Supplier. The result will display all the suppliers and their average account balance only in the Nations INDIA and FRANCE.

Likewise this query can be enhanced to take the data from various tables. Thus the number of joins in this query can be increased to produce results for various multi-join queries.

Example 1.2:(query with two joins)

```
Select S_Name, Avg (S_Acctbal)
From Nation, Supplier, Region
Where R_Regionkey = N_Regionkey And N_Nationkey =
S_Nationkey
Group By S_Name Having
Set(N_Name) Contained By {India, France}
```

Example 1.3:(query with three joins)

```
Select S_Name, Avg (S_Acctbal)
From Lineitem, Nation, Supplier, Region
Where N_Regionkey = R_Regionkey And N_Nationkey =
S_Nationkey And
S_Suppkey = L_Suppkey
Group By S_Name Having
Set(N_Name) Contained By {India, France}
```

Example 1.4:(query with four joins)

```
Select S_Name, Avg (S_Acctbal)
From Lineitem, Orders, Nation, Supplier, Region Where
R_Regionkey = N_Regionkey And N_Nationkey =
S_Nationkey And
S_Suppkey = L_Suppkey And
L_Orderkey = O_Orderkey And
L_Returnflag = R
Group By S_Name Having
Set(N_Name) Contained By {India, France}
```

Example 1.5:(query with five joins)

```
Select S_Name, Avg (S_Acctbal)
From Customers, Lineitem, Orders, Nation, Supplier,
Region
Where R_Regionkey = N_Regionkey And N_Nationkey =
S_Nationkey And
S_Suppkey = L_Suppkey And L_Orderkey = O_Orderkey
And O_Custkey = C_Custkey
Group By S_Name Having
Set(N_Name) Contained By {India, France}
```

The results on the TPC-H data set under different Multi-join complex queries using our proposed approach and existing Bitmap approach [15] are specified in Table 2. The corresponding graph for these results is shown in Fig. 2.

Table 2. Execution time of the multi-join queries

QTYPE-1		
No of Joins	**Bitmap**	**FBitmap**
One Join	46 secs	35 secs
Two Joins	47 secs	37 secs
Three Joins	48 secs	37 secs
Four Joins	50 secs	38 secs
Five Joins	52 secs	39 secs

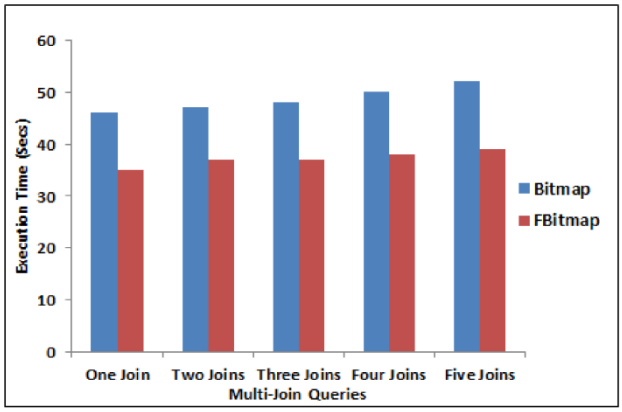

Fig. 2 Execution time of TPC-H dataset for various Multi-Join queries of querytype1

Query Type: 2

Here we consider the queries with different number of set predicates.

Example 2.1:(query with one set predicate)

Select S_Name, Avg(S_Acctbal)
From Customers, Orders, Lineitem, Supplier3, Nation, Region

Where N_Regionkey = R_Regionkey And S_Nationkey = N_Nationkey And L_Suppkey = S_Suppkey And O_Orderkey = L_Orderkey And C_Custkey = O_Custkey
Group By S_Name Having
Set(N_Name) Contained By {India,France}

Example 2.2:(query with two set predicates)

Select S_Name, Avg(S_Acctbal)

From Customers, Orders, Lineitem, Supplier3, Nation, Region

Where N_Regionkey = R_Regionkey And S_Nationkey = N_Nationkey And L_Suppkey = S_Suppkey And O_Orderkey = L_Orderkey And C_Custkey = O_Custkey

Group By S_Name Having

Set(R_Name) Contained By {Asia,Europe} And
Set(N_Name) Contained By {India,France}

Example 2.3:(query with three set predicates)

Select S_Name, Avg(S_Acctbal)

From Customers, Orders, Lineitem, Supplier3, Nation, Region

Where N_Regionkey = R_Regionkey And S_Nationkey = N_Nationkey And L_Suppkey = S_Suppkey And O_Orderkey = L_Orderkey And C_Custkey = O_Custkey

Group By S_Name Having

Set(R_Name) Contained By {Asia,Europe} And

Set(N_Name) Contained By {India,France} And
Set(C_Mktsegment) Contained By {Furniture, Building}

Example 2.4:(query with four set predicates)

Select S_Name, Avg(S_Acctbal)

From Customers, Orders, Lineitem, Supplier3, Nation, Region

Where N_Regionkey = R_Regionkey And S_Nationkey = N_Nationkey And L_Suppkey = S_Suppkey And O_Orderkey = L_Orderkey And C_Custkey = O_Custkey

Group By S_Name Having

Set(R_Name) Contained By {Asia,Europe} And

Set(N_Name) Contained By {India,France} And
Set(C_Mktsegment) Contained By {Furniture, Building} And

Set(L_Shipmode) Contained By {Ship,Rail}

Example 2.5:(query with five set predicates)

Select S_Name, Avg(S_Acctbal)

From Customers, Orders, Lineitem, Supplier3, Nation, Region

Where N_Regionkey = R_Regionkey And S_Nationkey = N_Nationkey And L_Suppkey = S_Suppkey And O_Orderkey = L_Orderkey And C_Custkey = O_Custkey

Group By S_Name Having

Set(R_Name) Contained By {Asia,Europe} And

Set(N_Name) Contained By {India,France} And
Set(C_Mktsegment) Contained By {Furniture, Building} And

Set(L_Shipmode) Contained By {Ship,Rail} And
Set(O_Orderpriority) Contained By { 2-High, 3-Medium}

Example 2.1 to 2.5 specifies the same query with different number of set predicates. The results on the TPC-H data set for the above query with

different number of set predicates using our proposed approach and existing Bitmap approach [15] are specified in Table 3. The corresponding graph for this result is shown in Fig. 3. From Fig.2 and Fig.3, it shows that Filtered Bitmap Index with multi-join multiple set predicates approach is more efficient than existing Bitmap algorithm [15] for processing multi-join query with multiple set predicates.

Table 3. Execution time of queries with multiple set predicates

QTYPE-2		
No of Set Predicates	**Bitmap**	**FBitmap**
One Predicate	53 secs	41 secs
Two Predicates	55 secs	42 secs
Three Predicates	58 secs	44 secs
Four Predicates	60 secs	45 secs
Five Predicates	62 secs	46 secs

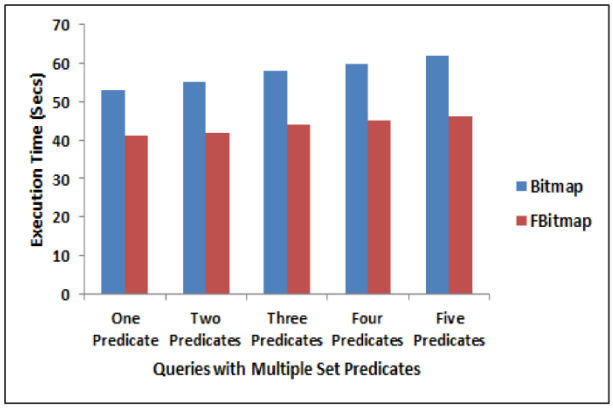

Fig. 3 Execution time of TPC-H dataset under multiple set predicates for querytype2

B. Experiments over WorldCup98 data:

The efficiency of our proposed algorithm is proved by using benchmark dataset worldcup-98 which is gathered from the internet site http://ita.ee.lbl.gov/html/contrib/ WorldCup.html.

The WorldCup98 data set comprise 1,352,804,107 tuples, which match to all the access pursuit eminent to the 1998 World Cup site between April 30, 1998 and July 26, 1998. Each tuple has data such as the instance of the request, the type of the requested file, the file size, the server that handleless the request, the client identifier (which maps to an IP address) and so on.

Query Type: 3

To discovery the sum of collection for case who had visited in two sequential days- July 18th, July 19th and accession all these file types HTML(1), JPG(2), and GIF(3).

Select Clientid, Sum(Bytes)
From Clients
Group By Clientid Having
Set(Date) Contain {0718,0719} And
Set(Type) Equal {1,2,3}

It identifies the clients who visited in both days July 18th, July 19th and the file types are equal to 1, 2 and 3.

The keyword CONTAIN represents a super set relation, i.e., the set variable SET(date) is a superset of {0718,0719}. The keyword EQUAL represents the equal relationship in set theory.

Results: The event on the WorldCup98 data set with opposite data sizes for the above query are shown in Table 4 with Fig. 4. We observed that the significant performance gains of the proposed method on billion-tuple dataset. The number of tuples is changed as 25, 50, 75 and 100 percent of the primary data set. From Fig.4, it shows that our proposed Filtered Bitmap Index with multi-join multiple set predicates approach is more efficient than existing Bitmap algorithm [15].

Table 4. Execution time with different data sizes for querytype3

QTYPE-3		
Data Size	**Bitmap**	**FBitmap**
25%	156 secs	102 secs
50%	160 secs	105 secs
75%	162 secs	109 secs
100%	165 secs	111 secs

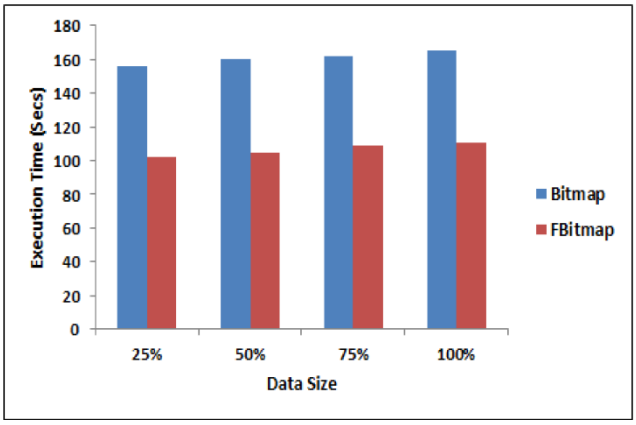

Fig. 4 Execution time of querytype3 under different data sizes for worldcup98 dataset

V. CONCLUSIONS

This paper presents an efficient algorithm Filtered Bitmap Index with multi-join multiple set predicates processing approach which is used for processing multi-join queries with multiple set predicates. The query with multiple set predicates allows simple syntax for complex query. This projected algorithmic rule has the good saving of disc accessing and the procedure time was attenuate by reduction the bit of iterations. In this algorithm the groups and the related sets are prescribed accordant to the query necessary that results will speeds up the query process. it increases the number of big accumulation storage for conclusion support applications, with efficiency execution aggregate queries are comely progressively important. The experimental results show that our proposed Filtered Bitmap Index with multi-join multiple set predicates approach is more efficient than the existing Bitmap algorithm for processing multi-join queries with

multiple set predicates. In future study our approach can be enhanced to tackle the problems of processing the queries in cloud environment also.

REFERENCES

1. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ, "A scalable continuous query system for internet databases", *In: Proc. of SIGMOD*, pp.379–390, 2000.
2. J. Albrecht, W. Hümmer, W. Lehner, L. Schlesinger, "Query Optimization By Using Derivability In a Data Warehouse Environment", *In: Proc. of the 3rd ACM international workshop on Data warehousing and OLAP, DOLAP*, pp. 49-56, 2000.
3. R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware", *Computer and System Sciences*, vol. 66, no. 4, pp. 614-656, 2003.
4. Panos Kalnis, Dimitris Papadias, "Multi-query optimization for on-line analytical processing", Information Systems, Volume-27,Issue 5, July 2003.
5. I.F. Ilyas, W.G. Aref, and A.K. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases", *VLDB J.*, vol. 13, no. 3, pp. 207-221, 2004.
6. Alaa Aljanaby, Emad Abuelrub, Jordan and Mohammed Odeh, "A Survey of Distributed Query Optimization", *The International Arab Journal of Information Technology*, Vol. 2, No. 1, January 2005.
7. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-so-Foreign Language for Data Processing", *In: Proc. ACM SIGMOD International Conference Management of Data*, pp. 1099-1110, 2008.
8. Javier Tuya, José Suárez Cabal and Claudio de la Riva, "Full predicate coverage for testing SQL database queries", Software Testing, Verification and Reliability, Vol. 20, No. 3, September 2010.
9. Pawan Meena, Arun Jhapate & Parmalik Kumar, "Framework for Query Optimization", *International Journal of Computer Science and Information Security*, Vol. 9, No. 10, October 2011.
10. Hui Zhao, Shuqiang Yang, Zhikun Chen, Songcang Jin, Hong Yin and Long Li, "MapReduce model-based optimization of range queries", In: Proc. of the 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012.
11. Swathi Kurunji, Tingjian Ge, Benyuan Liu, Cindy X. Chen, "Communication Cost Optimization for Cloud Data Warehouse Queries", In: Proc. of the IEEE 4th International Conference on Cloud Computing Technology and Science, 2012.
12. Davide Martinenghi and Marco Tagliasacchi, " Cost-Aware Rank Join with Random and Sorted Access", IEEE Transactions On Knowledge And Data Engineering, Vol. 24, No. 12, Dec 2012.
13. P. Arpitha, "Query Optimization in Data Warehouse", International Journal of Engineering Research & Technology, Vol. 2, Issue 8, 2013.
14. Swati Jain and Paras Nath Barwal, "Performance Analysis of Optimization Techniques for SQL Multi Query Expressions over Text Databases in RDBMS", Published in the International Journal of Information & Computation Technology, Volume 4, No. 8, 2014.
15. Chengkai Li, Member,IEEE, Bin He, Ning Yan, Muhammad Assad Safiullah "Set Predicates in SQL: Enabling Set-Level Comparisons for Dynamically Formed Groups", IEEE Transactions on Knowledge and Data Engineering , Vol. 26, No. 2, Feb 2014.
16. Jayant Rajurkar1 T. Khan2, "A System for Query Processing and Optimization in SQL for Set Predicates using Compressed Bitmap Index", International Journal for Scientific Research & Development, Vol. 3, Issue 02, 2015.
17. A.Regita Thangam and S.John Peter, "An Extensive Survey on Various Query Optimization Techniques", International Journal of Computer Science and Mobile Computing, Volume-5, Issue- 8, August 2016.
18. A.Regita Thangam and S.John Peter, "Efficient Processing and Optimization of Queries with Set Predicates using Filtered Bitmap Index", International Journal of Computer Sciences and Engineering, Volume-5, Issue-11, Nov 2017.
19. A.Regita Thangam and S.John Peter, "Efficient Processing of Queries with Set Predicates using Reduced Function based Approach", Journal of Emerging Technologies and Innovative Research, Volume-6, Issue-2, Feb 2019.
20. A.Regita Thangam and S.John Peter, "A Comparative Study of Query Processing and Optimization Techniques", In: Proc. of the International Conference on Recent Trends in Advanced Computing and its Applications, pp.74, Feb 2019.
21. TPC-H : http://www.tpc.org/tpc_documents_ current_ versions/ pdf/tpc-h _v2.17.3.pdf

AUTHORS PROFILE

Mrs. A. Regita Thangam is working as Assistant Professor in St.Xavier's College, Palayamkottai, Tamil Nadu, India. She is doing Ph.D. (Part Time) in Computer Applications at M.S. University, Abishekapatti, Tirunelveli. She earned his M.C.A. degree from M.S. University, Tirunelveli. She also earned his M.Phil from Alagappa University, Karaikudi. She has published research papers in many International journals and conference proceedings.

Dr. S. John Peter earned his M.Sc. and M.Phil. from Bharathidasan University, Tiruchirappalli. The M.S. University, Tirunelveli awarded him Ph.D. degree in Computer Science for his research in Data Mining. He is the Head of the department of computer science, and the Director of the computer science research centre, St.Xavier's college (Autonomous), Palayamkottai, Tirunelveli. The M.S. University, Tirunelveli has recognized him as a research guide. He has published research papers in International, National journals and conference proceedings. He has organized Conferences and Seminars at National level.