American-Eurasian Journal of Scientific Research 11 (5): 390-397, 2016
ISSN 1818-6785
© IDOSI Publications, 2016
DOI: 10.5829/idosi.aejsr.2016.11.5.22958

Query Processing and Optimization Using Set Predicates

Rhia Mariam George and A. Ronalad Doni

Department-MCA, Faculty of Computing,
Sathyabama University, Chennai-600119, Tamilnadu, India

Abstract: The scalar-level implication in SQL becomes progressively important to support a new group of operation that needs set-level contrast semantics. That is matching a tuples group with multiple values. Complicated queries of SQL constructed using scalar-level operation are frequently formed to get even simplest set-grade semantics. This type of queries is not only challenging to write and also difficult with regard to database engine optimization, so they may result in expensive evaluation. To overcome this problem, in this paper we suggests to become greater SQL with a new predicates type, *set predicates*, to come carrying out the as an alternative easily understood semantics and accordingly ease the efficient and direct support. Here we suggested two methods of processing set predicates in this study: an approach based on bitmap index; another approach based on aggregate function. Aggregate functions find their usage for searching tuples which satisfy certain aggregate quality over a group of tuples (in contrast to just calculating an aggregate quality upon a group of tuples). The approach based on bitmap index might be used for eliminating the need for scanning and handling the complete data set (like a table) that ends in significantly speeding up the required query processing. Furthermore, we have devised a probabilistic method that is based on histogram for estimating set predicate selection, in view of optimizing queries having multiple predicates. Also, the experiments have verified the methods accuracy as well as effectiveness of optimizing queries.

Key words: Bitmap index · Set predicate · Data warehousing · OLAP · Query processing and Evaluation

INTRODUCTION

In the recent days, there has been considerable demand for querying data in OLAP and data warehouse applications with semantics of pair-level comparison. Consider the case of an institution or a company that seeks candidates for a few jobs with a required set of mandatory skills. The institution or company would search their database of resume. Skills pertaining to each candidate which is a collection of values will be compared with the conditional skills required. Such collections will be formed dynamically. This procedure of group level comparisons may be executed with the use of presently available SQL semantics and syntax without suggested system [1]. In case the group level contrasts are performed using SQL syntax that is presently available, the resulting query can, at times, be increasingly complicated; ending up with consuming too long a time than necessary, for processing the query. Such complicated query gets too tough for users to devise and results in far too expensive an evaluation [2].

Example 1: With a view of determining candidates having skills Java and PHP, you can write a query as under: At the end of grouping, a dynamic group of values about attribute proficiency will be created for each specific id and groups with matching SET (skill) that contain Java as well as PHP will be returned.

Example 2: The set predicates concept may be described over multiple properties. Take the case of any online advertisement for example. Consider table schema as Site Statistics (advertiser, website, CTR). The following query could be used by a marketing strategist for finding websites publishing ads on behalf of ING under gretaer than 2 percent of tick-through rate (CTR) as well as dont publish advertisements for HSBC so far: In this given example, the primary set predicate includes two properties while the other set predicate would use the negation, namely, CONTAIN. It may be noted that we employ >0.02 for representing a condition that is range-based (CTR>0.02). In many cases, semantics of group-level comparisons may be stated by using the current available

Corresponding Author: Rhia Mariam George, Department-MCA, Faculty of Computing, Sathyabama University, Chennai-600119, Tamilnadu, India.

given SQL syntax without suggested extension. But then, the queries resulting will be increasingly complicated than required. A significance to be noted is that complicated queries can be difficult to create for users. More important, such complicated queries may prove to be tough for DBMS in optimizing and this leads to expensive evaluation that is unnecessary [2]. The query plans resulting from these may involve certain multiple inner queries involving grouping as well as set processes. The suggested syntax with set predicates empowers direct stating of group-level comparisons within SQL and this makes formulation of query easy. It also fosters effective support to such queries. Also, with threshold constraint concept, a special kind of query like iceberg query normally returns only a meager percentage of definite groups in the form of output. Due to the little result set, Iceberg queries may be potentially responded faster even in the case of very huge data set [3]. Certain approaches and / or database systems that are presently available are not able to absorb the benefit of the feature of the iceberg query. All the currently used relational database structure (e.g., MySQL, DB2, Sybase, Postgre SQL, Oracle, SQL Server and column-based databases LucidDB, Vertica, MonetDB) are making use of common algorithms of aggregation [4, 5], for answering iceberg queries first by aggregation of every tuple and after that evaluation of the HAVING condition for selecting the iceberg output. Multipass aggregating algorithms can be employed in the case of large data set wherein the complete aggregate output would not fit in the memory. Most of the present methods for query optimization related to iceberg queries [6] may be classified as the methods based on tuple-scan and they need a minimum of one table for reading the data from the disk. These focus on decreasing number of the passes in the case of data size being large [2]. No method has leveraged the quality of iceberg queries effectively for productive processing. I t takes a very long time for responding to iceberg queries meant for such scheme based on tuple-scan, particularly in the case of a considerably large table.

Relevant Work: These days, a lot of database management schemes offer description of features including a collection of values like nested table found in Oracle and also in SET data type as in MYSQL. Data storage and depiction is not needed for Set predicates as standard DBMS includes them. In practical applications, in accordance with requirement of query collections, matching sets are normally created dynamically. It is possible for users to create set level contrasts dynamically without having any limitation due to schema of database fir set predicates. Cross feature set level correlation is also allowed. In [7, 8, 9], collection of variables and related set concepts were proposed as extension of SQL for allowing correlation of the multiple aggregate activities upon the same assembling condition. This study focuses primarily on data processing with the use of condensed bitmap index as well as prediction of the sets. Apart from that, Set predicate also gets related with relational division and universal quantification that happen to be robust in the analysis of relationships of many-to-many type. One sample query for universal quantification is finding out students who have engaged in every computer science course required for graduation. It becomes a special kind of set predicates having CONTAIN operator upon all values for a feature in the table, Courses. In contrast to this, the suggested set predicates permit dynamic formation of sets through GROUPBY and also back CONTAINED BY as well as EQUAL, apart from CONTAIN [2]. FASTBIT and the RIDBIT methods were proposed by Elizabeth ONeil at al. FastBit is a research tool that was developed to study and analyze the effect of compression methods on bitmap indexes and it has found its usage in many scientific exercises. In this method, table data are organized into columns and rows, wherein every table gets partitioned vertically, each column gets stored in separate file, while each partition will typically contain a number of rows. The Bitmap indexes get continuously applied without being partitioned into bit sections as it is done in RIDBIT method. The index that is being employed in this paper happens to be Word-aligned hybrid (WAH) condensed primary bitmap index. Whereas in case of FastBit tool, the bitmap creates all value of complete index for any one individual on the memory prior to writing index file. Bin He *et al*., (2012) illustrated the bitmap indexes properties and created a powerful and very effective bitmap index trimming method for query processing. The technique based on Bitmap index pruning helps remove the need for scanning and refining the complete data set (a table), hence it results in very fast processing of queries. This method is found to be highly efficient compared to other present algorithms that have been normally used in the recent databases. Opportunities to compute queries very efficiently with the use of condensed bitmap index are increased by examining these bitmap indices qualities. Doing pair wise bitwise AND procedures amid bitmap vectors related to all necessary features is one simple way for calculating query with the use of bitmap indices. This method is not so effective as the outcome of number denoting bitmap vectors pertaining to every attribute is huge and a large amount of these procedures mare not needed. Different schemes of compression are found to have been created for bitmap index. The two crucial schemes of compression which is possible to be exercised on any column and that is possible to be employed for processing queries without decompression are Bitmap Code based on byte alignment Code (BBC) [10] and Word-Aligned Hybrid (WAH) [11]. The evolution of further bitmap condensation strategies and coding methods has still broadened bitmap index applicability [12]. In the recent days, it has become possible to apply this on all numeric features and text features. It is also very effective for warehouse query processing and OLAP. Anyhow, bitmap index so far has not been leveraged effectively in the existing works for processing iceberg queries. In our study, we have evolved algorithms for query processing with the use of bitmap.

Proposed Work: The introduced compact syntax pertaining to set predicates empower direct display of group-level contrasts in SQL that not only assists effective support of these queries but also simplifies the query formulation. We have given one method based on aggregate function and another method based on bitmap index. Lastly, we have promoted one probabilistic method that is histogram-based, for estimating the selection of any set predicate. This research has been related mostly with set-valued features and group restraint joins. In contrast, the suggested answer to set predicates is found to have many crucial benefits:

- Opposed to set-valued features that involve significant problems with re-devising database storage of specific data type about group, set predicate needs no alteration in data depiction or storage engine and hence it can directly be accommodated into any of the standard relational databases.
- By making use of set predicates, it is possible for users to form set-level contrasts dynamically based on the query requirements without any constraints created due the database schema. In contrast, in the case of set-valued features, it is possible to pre-define sets statically in the design phase of schema. So group-level comparisons become possible only on group-valued aspects and non-group query terms were not supported easily on group-valued aspects.

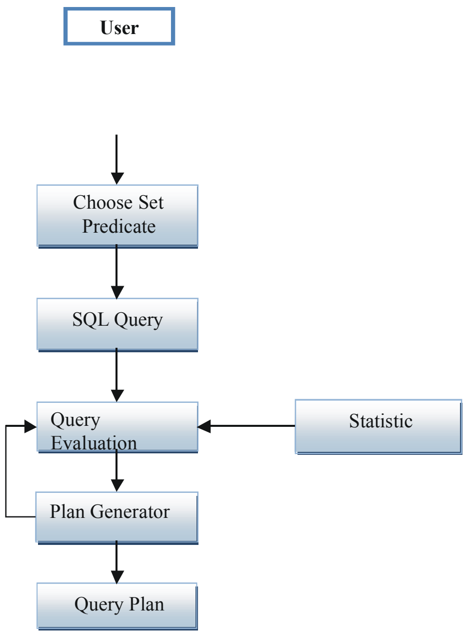


Fig. 1: Overall Architecture

- It permits cross-aspect group-level comparison that will not be backed by group-valued aspects.

Creation of CRN: Rather than referring to the complete course itself, a course reference indication number normally points to some particular portion of any course. Often times, classes that have a large number of students, say, several hundred, get divided into classes that are smaller with 20 or 30 students. Such smaller sections can be indicated by using course reference indication numbers that usually have a length of five digits. Course reference numbers can be displayed in different locations in different colleges.

Set Predicates: SQL syntax has been extended for supporting set predicates. Set predicate gels well with GROUP BY and also HAVING clauses, as it compares a set of tuples to a group of values. Particularly, in the case of HAVING clause, any Boolean expression upon multiple common aggregate predicates and also set predicates is found, linked by using logical operators such as ANDs, Ors and NOTs.

Multi Attribute Gathering: In the event of many column names occurring inside any GROUP BY clause, a resulting table may be split into divisions inside divisions. Consider the example where, if you happen to mention column names indicating district, year and region in GROUP BY section.

Multi Attribute Set Predicate: Comparison of sets described on multiple aspects is also allowed by query syntax. Clauses are mentioned below:

In: This is for determining whether any particular values are matching with any of the values found in the list.

Contains: Although it has similarity to free text it differs in the fact that it can take only one keyword for matching with a record. In case it is necessary to join another word, it is possible by using AND or OR operators.

- Free Text: Free text is one predicate that may be used for searching columns consisting data type based on characters. While it does not match the exact word it matches meaning of words mentioned in search stipulation.
- Multi Predicate Set Operation: A query that contains manifold set predicates may be supported with the use of Boolean Operators such as AND, NOT and OR.
- Aggregate Expression: Inherent aggregates are the aggregate functions devised by database server like SUM, COUNT and AVG. Such aggregates function only with inherent data types like FLOAT and INTEGER.

Query Assessment Approach

Aggregate Function-Based Method: The introduced syntax that propels set predicates' semantics, so that a query plan that is aware about set predicates may be highly efficient by merely scanning of any table and handling the tables tuples in sequential manner. The vital factor in this direct method is performing grouping as well as group-level comparison in sync, under one-pass repetition of tuples. This idea is similar to processing normal aggregate functions in sync with grouping. Therefore we develop a technique that manages set predicates like aggregate functions.

Method Based on Bitmap Index: The method based on bitmap index just requires bitmap indices over individual aspects. Based on individual-aspect indices, the easy data pattern and bitmap processes make it easy to integrate different processes in any query, inclusive of dynamic assembling of the tuples and group-level comparisons.

Probabilistic Method Based on Histogram: The occurrence frequency for all distinct values in any data set can be measured by a histogram. Query optimizer evaluates a histogram for column values found in primary key column related to statistics object, by choosing column values using statistical inspection of the rows by executing a complete scan of each row in view or the table. In case the histogram has been generated through sampled group of rows, then the sums of number of the rows and the number of definite values can be assessed and they need not be any whole integers as well.

The Histogram Steps Are:

- RANGE_HI_KEY
  It displays the histogram step upper bound value.

- RANGE_ROWS
  It shows the number of rows from the sample that drop within a step of histogram, not including the upper bound.

- EQ_ROWS
  It shows the rows from sample that are same in value to the histogram step upper
  bound.

- DISTINCT_RANGE_ROWS
  It shows the number of different values within the histogram step, not including the upper bound.

- AVG_RANGE_ROWS
  It shows duplicate values average number within histogram step, not including the upper bound value ((RANGE_ROWS / DISTINCT_RANGE_ROWS for DISTINCT_RANGE_ROWS > 0).

Statistics: It is possible to collect statistics by inspecting all the rows in a table or through sampling any huge table. In the event of sampling being used, or if statistics are considered obsolete, the statistics that has been furnished on the page will not exhibit the tables true state. Statistics that have been revealed happen to be read-only. Such information can be used by query optimizer for creating the most fitting query plan that is possible.

Options:

- Table Name
  It shows table name described by statistics.

- Statistics Name
  It shows the database object name where statistics are stored. Unrelated statistics to an index, that were formed by Microsoft SQL server have name starting with WA Sys.

- Statistics for
  It shows the statistics object name.

- Updated
  It shows the date the present statistics were created.

- Rows
  It shows the rows in the table

- Rows Sampled
  Shows the no of rows inspected to generate the statistics.

- Steps
  The table rows are split to groups for histogram of statistics. This is the number of groups that were generated.

- Density
  An index that has more number of duplicates has more density.

A distinctive index has little density.

- Average Key Length
  It shows the each row's average length.

- String Index
  Yes specify that the statistics contain a index of string summary to support result set sizes estimation for LIKE conditions.

- Data for Columns:
- All Density
  It shows the thickness for columns combination that are listed in the section of Columns.

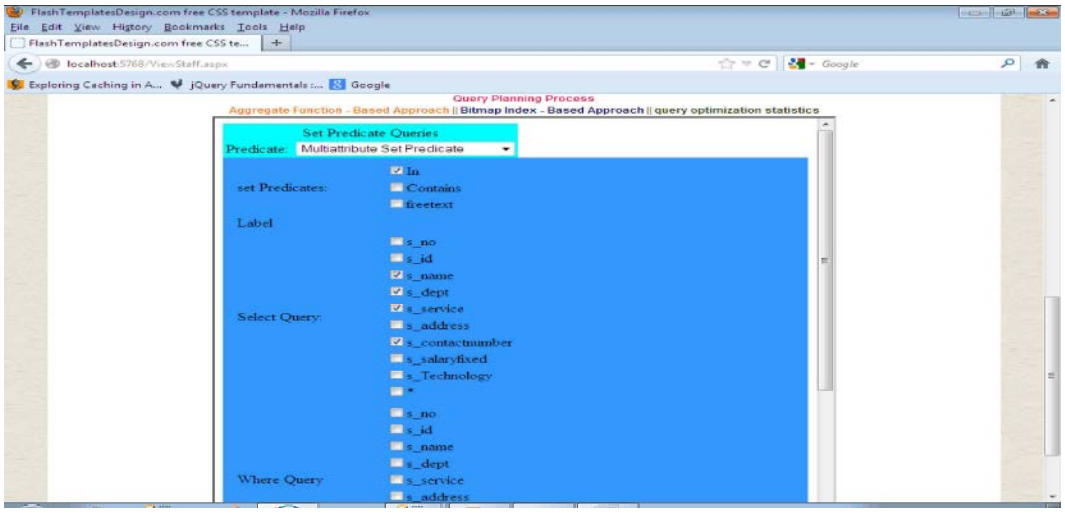
- Average Length
  It shows the average length of columns combination that are listed in the section of

- Columns
  It shows the columns described by the Average Length and the All Density fields.

RESULT AND DISCUSSION

This paper presents an efficient algorithm for query processing using compressed bitmap indices and Aggregate function based technique. Using these algorithms we process the set predicate for retrieve data from large number of datasets.

The Above Figure 2 and Figure 3 shows the result from staff details table using the In clause set predicate. And also it shows the dynamically formed query on screen that depends on the input given by the user. It result the values based on the query.


FlashTemplatesDesign.com free CSS template - Mozilla Firefox
File Edit View History Bookmarks Tools Help
FlashTemplatesDesign.com free CSS te...
localhost:5768/ViewStaff.aspx
Google
Exploring Caching in A... jQuery Fundamentals :... Google
Query Planning Process
Aggregate Function - Based Approach || Bitmap Index - Based Approach || query optimization statistics

Set Predicate Queries
Predicate: Multiattribute Set Predicate

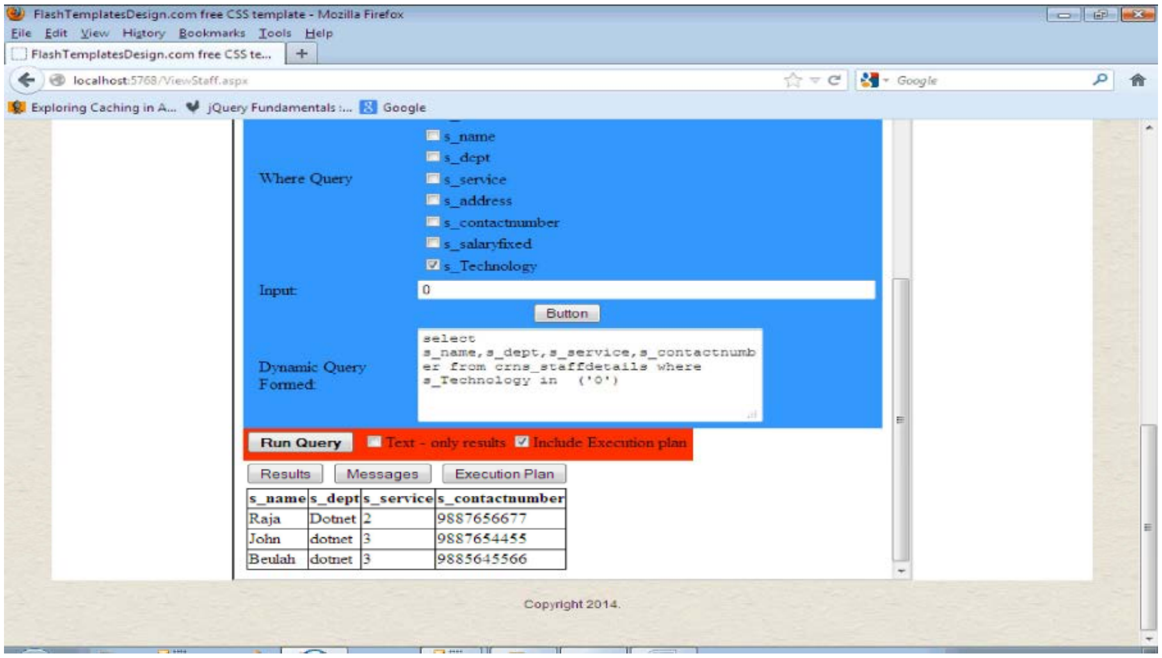
set Predicates: [✓ In] Contains freetext

Label

Select Query: s_no s_id [✓ s_name] [✓ s_dept] [✓ s_service] s_address [✓ s_contactnumber] s_salaryfixed s_Technology *

Where Query s_no s_id s_name s_dept s_service s_address

Fig. 2: Using in clause Set Predicates


FlashTemplatesDesign.com free CSS template - Mozilla Firefox
File Edit View History Bookmarks Tools Help
FlashTemplatesDesign.com free CSS te...
localhost:5768/ViewStaff.aspx
Google
Exploring Caching in A...
jQuery Fundamentals :...
Google
Button
Run Query
Results
Messages
Execution Plan
Copyright 2014.

Where Query
☐ s_name
☐ s_dept
☐ s_service
☐ s_address
☐ s_contactnumber
☐ s_salaryfixed
☑ s_Technology

Input: 0

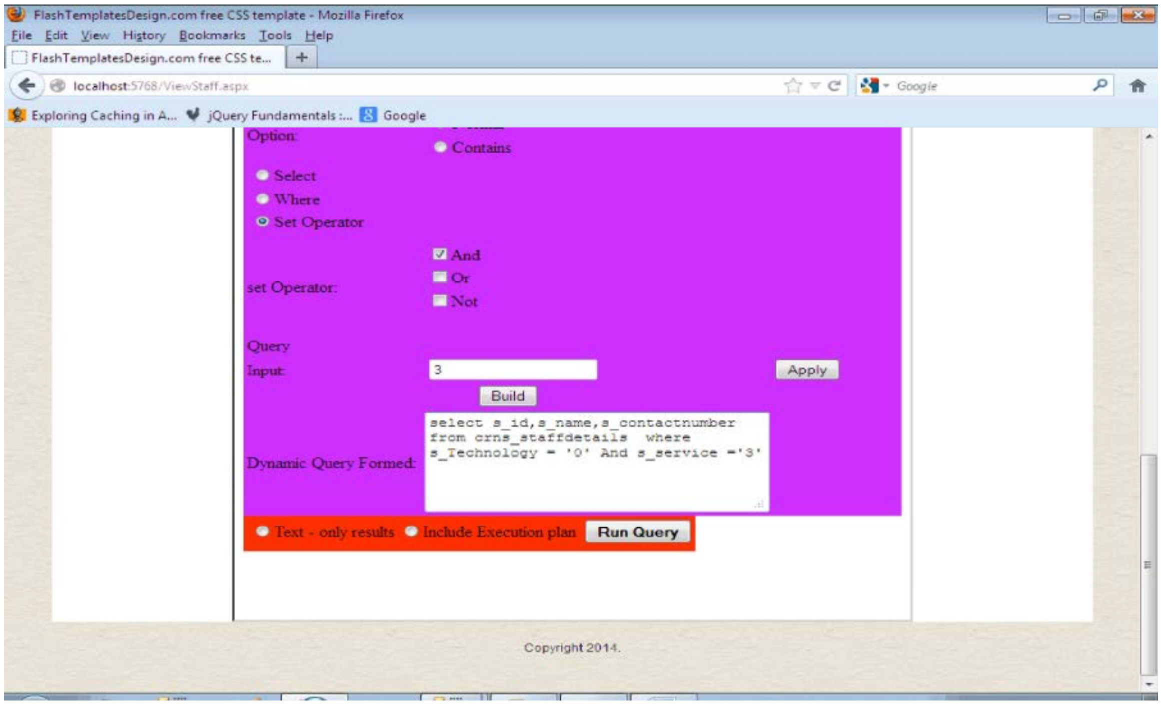
Dynamic Query Formed:

```
select
s_name,s_dept,s_service,s_contactnumb
er from crns_staffdetails where
s_Technology in  ('0')
```

☐ Text - only results ☑ Include Execution plan

s_name	s_dept	s_service	s_contactnumber
Raja	Dotnet	2	9887656677
John	dotnet	3	9887654455
Beulah	dotnet	3	9885645566

Fig. 3: Queries Formed Dynamically and Shows the Results


FlashTemplatesDesign.com free CSS template - Mozilla Firefox
File Edit View History Bookmarks Tools Help
FlashTemplatesDesign.com free CSS te...
localhost:5768/ViewStaff.aspx
Google
Exploring Caching in A...
jQuery Fundamentals :...
Google
Apply
Build
Run Query
Copyright 2014.

Option:
○ Contains
○ Select
○ Where
◉ Set Operator

set Operator:
☑ And
☐ Or
☐ Not

Query Input: 3

Dynamic Query Formed:

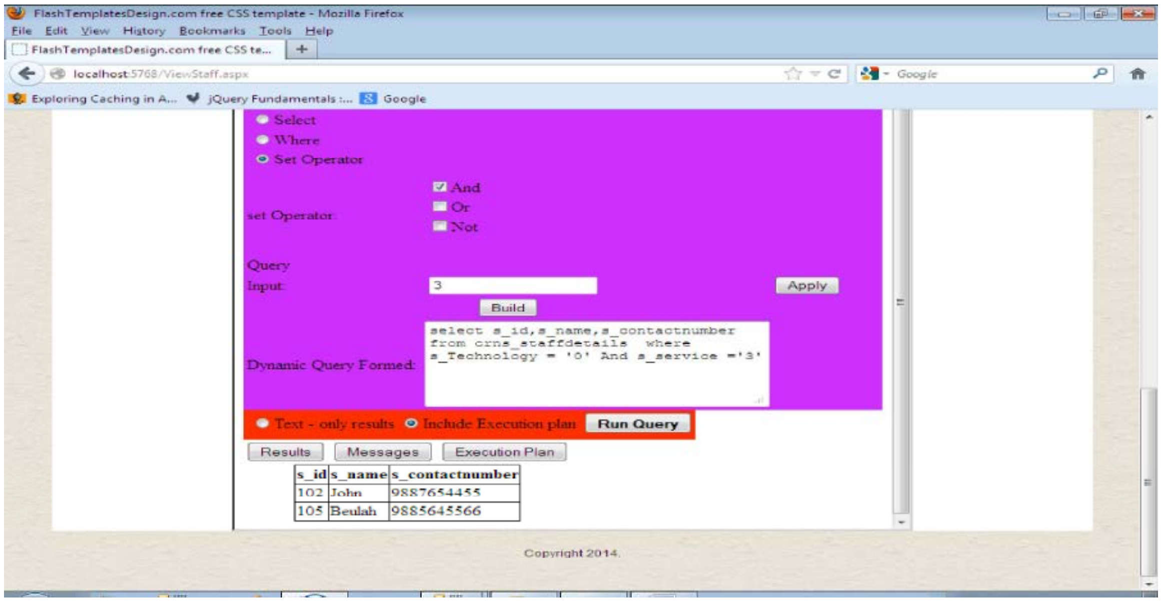
```
select s_id,s_name,s_contactnumber
from crns_staffdetails  where
s_Technology = '0' And s_service ='3'
```

○ Text - only results ○ Include Execution plan

Fig. 4: Using and clause Set Operator

The above Figure 4 and 5 shows the results using and operator. The And operator displays record from crns_staffdetail table in that both condition is true in given table, that record will be displayed in result place.

The Figure 6 shows comparison of query retrieved processing among existing techniques. In aggregate technique time required to retrieve records from datasets is 29611ms and Bitmax technique take around 12000ms to retrieve the records. In our proposed system time required to retrieve the records is much less than existing system.



FlashTemplatesDesign.com free CSS template - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost:5768/ViewStaff.aspx
Exploring Caching in A... jQuery Fundamentals :... Google
Build
Apply
Run Query
Results
Messages
Execution Plan
Copyright 2014.

( ) Select
( ) Where
(●) Set Operator

set Operator: [✓ And] Or Not

Query Input: 3

Dynamic Query Formed:

```
select s_id,s_name,s_contactnumber
from crns_staffdetails  where
s_Technology = '0' And s_service ='3'
```

Text - only results [✓ Include Execution plan]

s_id	s_name	s_contactnumber
102	John	9887654455
105	Beulah	9885645566

Fig. 5: Shows the results Based on and Operator

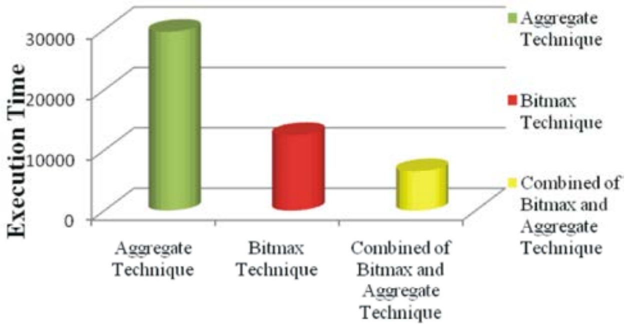


Fig. 6: Result Analysis

CONCLUSION

In the research, we have submitted an extensive analysis about Set Predicates for supporting group-level comparisons. These kinds of predicates, when joined into a group, permit choice of groups and group values that are formed dynamically. Two methods have been proposed by us, one based on aggregate function and the other, based on consolidated bitmap index, for processing of set predicates. The bitmap index was observed to have the following advantages:

- By warding off tuple-scan in tables with a large number of features, it saves disk access.
- By carrying out bitwise processes, it reduces computation time.The trouble with huge blank AND results may be removed with the use of effective vector sequence algorithm with precedence queues. Moreover, we have developed an optimization technique for further improving the general performance concerning the entire system.

REFERENCES

1. Bin, He and Hui-l Hsiao, 2012. Member IEEE, Ziyang Liu, Yu Huang and Yi Chen, Member, IEEE, Efficient Iceberg Query Evaluation Using Comressed Bitmap Index, IEEE Transactions on Knowledge and Data Engineering, 24(9).
2. Chengkai Li, 2014. 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, 26(2).
3. Olston, C., B. Reed, U. Srivastava, R. Kumar and A. Tomkins, 2008. Pig Latin: A Not-so-Foreign Language for Data Processing, Proc. ACM SIGMOD Intl Conf. Management of Data, pp: 1099-1110.

4. Melnik, S., A. Gubarev, J.J. Long, G. Romer, S. Shivakumar, M. Tolton and T. Vassilakis, 2011. Dremel: Interactive Analysis of Web- Scale Data Sets, Comm. ACM, 54: 114-123.
5. Wu, K., E.J. Otoo and A. Shoshani, 2006. Optimizing Bitmap Indices with Efficient Compression, ACM Trans. Database Systems, 31(1): 1-38.
6. Ioannidis, Y., 2003. The History of Histograms (Abridged), Proc. Intl Conf. Very Large Databases (VLDB).
7. Chatziantoniou, D. and K.A. Ross, 1996. Querying Multiple Features of Groups in Relational Databases, Proc. Int'l Conf. Very Large Databases (VLDB), pp: 295-306.
8. Chatziantoniou, D. and K.A. Ross, 1997. Groupwise Processing of Relational Queries, Proc. 23rd Int'l Conf. Very Large Databases (VLDB), pp: 476-485.
9. Chatziantoniou, D. and E. Tzortzakakis, 2009. Asset Queries: A Declarative Alternative to Mapreduce, ACM SIGMOD Record, 38(2): 35-41.
10. Mamoulis, N., 2003. Efficient Processing of Joins on Set-Valued Attributes, Proc. ACM SIGMOD Intl Conf. Management of Data, pp: 157-168.
11. Melnik, S. and H. Garcia-Molina, 2003. Adaptive Algorithms for Set Containment Joins, ACM Trans. Database Systems, 28(1): 56-99.
12. Stonebraker, M., D.J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E.J. ONeil, P.E. ONeil, A.Rasin, Tran and S.B. Zdonik, 2005. C-Store: A Colum- OrientedDBMS, Proc. Intl Conf. Very Large Data Bases (VLDB), pp: 553-564.