International Journal of Technology and Engineering System (IJTES)
Vol 7. No.5 2015 Pp. 470-477
©gopalax Journals, Singapore
available at : www.ijcns.com
ISSN: 0976-1345

Query Processing and Optimization Using Set Predicates

C.Saranya[1], B.Kamala[2]
[1]PG Student, [2]Assistant Professor
[1,2]Department of Computer Application, Sri Sai Ram Engineering College, Chennai.

ABSTRACT

The SQL is extended with set predicates for an important class of analytical queries, which otherwise would be difficult to write and optimize. It is designed in two query evaluation approaches for set predicates, including an aggregate function-based approach and a bitmap index-based approach. Observing the demand for complex and dynamic set level comparisons in database, so the concept of set predicates is used. The experiment is verified for its accuracy and effectiveness in optimizing queries.

Key Terms-Set predicates, grouping, data warehousing, OLAP, querying processing and optimization.

I. Introduction

In modernization of technological era, the amount of quality of the businesses in many application domains. In data has been exploding in many application domains. For instance, A Course Reference Number (CRN) is a unique identifier assigned to a specific class section at an educational institution. This is in contrast to a course number, which follows other conventions and is used to refer to the course itself, instead of a specific section of the course. They needs to retain long-term active data or even permanent are increasing. Data are accumulated and transformed to big dataset before they can be stored in a database. Big datasets can cause overhead to database performance issues. There is a high demand of querying data with the semantics of set level comparisons. Database performance is a crucial issue, which can decrease the ability of the DBMS to respond to queries quickly. Poor database performance cause negative consequences such as in financial, productivity and quality of the businesses in many application domains. Observing the demand for complex and dynamic set level comparisons in databases, we propose a concept of set predicate.

II. Related Work

Set-valued attributes provide a concise and natural way to model complex data concepts such as sets. Many DBMSs nowadays support the definition of attributes involving a set of values, for example, nested table in Oracle and SET data type in MySQL. For example, the “skill” attribute in Example 1 can be defined as a set data type. Set operations can be natively supported on such attributes. Query processing on set-valued attributes and set containment joins have been extensively studied [14] Although set-valued attributes together with set containment joins can support set-level comparisons, set predicates have several critical advantages:

1. Unlike set-valued attributes, which bring hassles in redesigning database storage for the special set data type, set predicates require no change in data representation and storage, and thus can be incorporated into standard RDBMS.
2. In real-world applications, groups and corresponding sets are often dynamically formed according to query needs. For instance Example 2, the monthly ratings of each department form a set. In a different query, sets may be formed by ratings of individual employees. With set predicates, users can dynamically form set-level comparisons with no limitation caused by database schema. On the contrary, set-valued attributes cannot support dynamic set formation because they are predefined at schema definition phase and set-level comparisons can only be issued on such attributes.
3. Set predicates allow cross-attribute set-level comparison. For instance, sets are defined over advertiser and CTR together in Example 3. On the contrary, a set-valued attribute can only be defined on a single attribute in many implementations, thus cannot capture cross-attribute associations. Implementations such as nested table in Oracle allow sets over multiple attributes but do not easily support set- level comparisons on such attributes. Set predicate is also related to universal quantification and relational division [12], which are powerful for analyzing many-to-many relationships. An example universal quantification query is to find the students that have taken all computer science courses required to graduate. It is a special type of set predicates with CONTAIN operator over all the values of an attribute in a table, for example, Courses. By contrast, the proposed set predicates allow sets to be dynamically formed through GROUP BY and support CONTAINED BY and QUAL, in addition to CONTAIN. The SEQUEL 2 language (an extension of the original SEQUEL) for SYSTEM R proposed a special SET function, for comparing a set of attribute values with the result of a subquery [4]. The proposed comparison operators include CONTAINS, =, and their negations. Furthermore, these operators can be used in comparing the results of two.

III. III. EXISTING SYSTEM

The semantics of set-level comparisons in many cases can be expressed using current SQL syntax without the proposed extension. However, resulting queries would be more complex than necessary. One consequence is that complex queries are difficult for users to formulate. More importantly, such complex queries are difficult for DBMS to optimize, leading to unnecessarily costly evaluation. The resulting query plans could involve multiple sub queries with grouping and set operations.

Disadvantage:

- Queries are generate without any conditions.
- It needs lengthy queries , so it seems to be a complex task.
- Wastage of resource.
- Retrieval is complex

IV. Proposed System

The proposed concise syntax of set predicates enables direct expression of set-level comparisons in SQL, which not only makes query formulation simple but also facilitates efficient support of such queries. We developed two approaches to process set predicates: Aggregate function and Bitmap index-based approach. Finally, we developed a histogram-based probabilistic method to estimate the selectivity of a set predicate.

Set Predicates in SQL

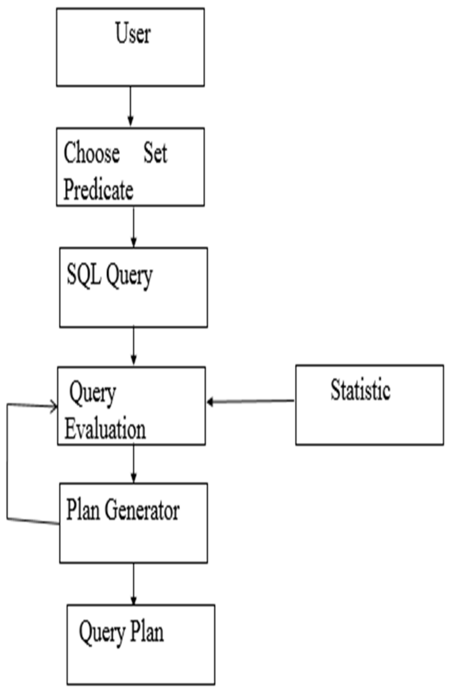


Fig.1. Predicate Architecture

Set Predicates

The SQL syntax is to support set predicates. Since a set predicate compares a group of tuples to a set of values, it fits well into GROUP BY and HAVING clauses. Specifically, in a HAVING clause there is a Boolean expression over multiple regular aggregate predicates and set predicates, connected by logic operators ANDs, ORs, and NOTs. The syntax of a set predicate is

SET(v1;...;vm) CONTAIN j CONTAINED BY j EQUAL fðv1 1;...;v1 mÞ;...;ðvn 1;...;vn mÞg, where vj i 2 DomðviÞ, i.e., each vj i is a literal value (integer, floating point number, etc.) in the domain of attribute vi. Succinctly, we denote a set predicate by ðv1;...;vmÞ op fðv1 1;...;v1 mÞ;...;ðvn 1;...;vn mÞg, where op can be ; , and =, corresponding to set operator CONTAIN, CONTAINED BY, and EQUAL, respectively. The architecture includes some modules:

- CRN’s Creation
- Set predicates
- Query Evaluation approach
- Aggregate Function
- Bitmap Index
- Histogram based Probabilistic Method
- Statistics

Module Description:

1. CRN’s Creation

A course reference number usually refers to a specific section of a course, rather than the whole course itself. Often, large classes with several hundred students are divided into smaller classes of 20 or 30; these smaller sections are indicated by course reference numbers, usually five digits long. Different colleges display course reference numbers in different places

2. Set Predicates

We extend SQL syntax to support set predicates. Since a set predicate compares a group of tuples to a set of values, it fits well into GROUP BY and HAVING clauses. Specifically, in a HAVING clause there is a Boolean expression over multiple regular aggregate predicates and set predicates, connected by logic operators ANDs, ORs, and NOTs.

General queries are:

i. Multi attribute Grouping

When several column names occur in a GROUP BY clause, the result table is divided into groups within groups. For example, if you specify column names for year, region, and district in the GROUP BY clause.

ii. Multi attribute Set Predicate

The query syntax also allows comparing sets defined on multiple attributes. They clauses are:

a. In

It determines whether a specified value matches any value in a list.

b. Contains

It is similar to the free text but with the difference that it take one keyword to match with the record. If you want to combine another word you can use AND, OROperator.

c. Free text

FFree text is a predicate used to search columns containing character based data type. It will not match exact word but the meaning of the words in the search condition.

d. Multi predicate Set Operation

A query with multiple set predicates can be supported by using Boolean Operators .i.e. AND, OR and NOT.

e. Aggregate Expression

Built-in aggregates are aggregate functions that are defined by the database server, such as AVG, SUM, and COUNT. These aggregates work only with built-in data types, such as INTEGER and FLOAT.

3. Query Evaluation approach

i. Aggregate Function based approach:

With the new syntax which brings forward the semantics of set predicates, a set predicate-aware query plan could potentially be much more efficient by just scanning a table and processing its tuples sequentially. The key to such a direct approach is to perform grouping and set-level comparison together, through a one-pass iteration of tuples. The idea resembles how regular aggregate functions can be processed together with grouping. Hence, we design a method that handles set predicates as aggregate functions.

ii. Bitmap Index based approach:

The bitmap index-based approach only needs bitmap indices on individual attributes. Based on single-attribute indices the simple data format and bitmap operations make it convenient to integrate various operations in a query, including dynamic grouping of tuples and set-level comparisons.

4. Histogram based Probabilistic Model

A histogram measures the frequency of occurrence for each distinct value in a data set. The query optimizer computes a histogram on the column values in the first key column of the statistics object, selecting the column values by statistically sampling the rows or by performing a full scan of all rows in the table or view. If the histogram is created from a sampled set of rows, the stored totals for number of rows and number of distinct values are estimates and do not need to be whole integers

The Histogram Steps are:

RANGE_HI_KEY

It displays the upper bound value of a histogram step.

RANGE_ROWS

It displays the number of rows from the sample that fall within a histogram step, excluding the upper bound.

EQ_ROWS

It displays the number of rows from the sample that are equal in value to the upper bound of the histogram step.

DISTINCT_RANGE_ROWS

It displays the number of distinct values within a histogram step, excluding the upper bound.

AVG_RANGE_ROWS

It displays the average number of duplicate values within a histogram step, excluding the upper bound value (RANGE_ROWS /

DISTINCT_RANGE_ROWS for DISTINCT_RANGE_ROWS > 0).

5. Statistics

Statistics can be collected by examining every row in the table or by sampling a large table. When sampling was used, or when statistics are out of date, the statistics presented in the page may not reflect the exact state of the table. The statistics displayed are read-only. This information is used by the query optimizer to create the best possible query plan.

Options:

Table Name

It displays the name of the table described by the statistics.

Statistics Name

It displays the name of the database object where the statistics are stored. Statistics unrelated to an index which were created automatically by Microsoft SQL Server have names beginning with _WA_Sys.

Statistics for

It displays the name of the statistics object.

Updated

It displays the date the current statistics were created.

Rows

It displays the number of rows in the table.

Rows Sampled

Displays the number of rows examined to create the statistics.

Steps

The rows of the table are split into groups for the statistics histogram. This is the number of groups that were created.

Density

An index that has a large number of duplicates has high density. A unique index has low density.

Average Key Length

It displays the average length of each row.

String Index

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

Data for Columns:

All Density

It displays the density for the combination of columns listed in the Columns section.

Average Length

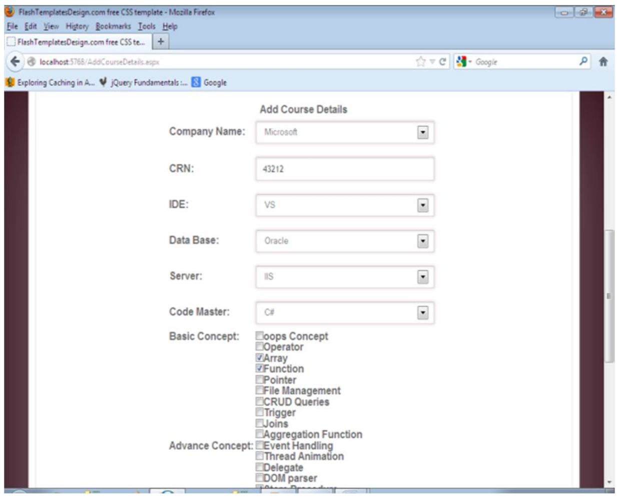
It displays the average length of the combination of columns listed in the Columns section.

Columns

It displays the columns described by the All Density and the Average Length fields.

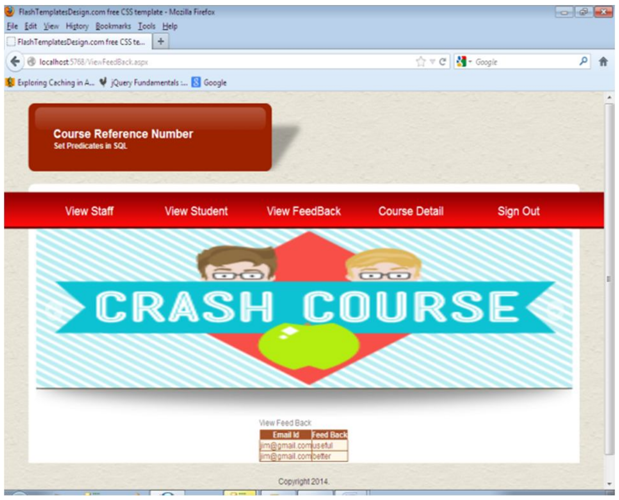
V. Implementation and Result

The conducted experiments on both query processing algorithms and query optimization techniques and compared the performance of three methods in evaluating set-level comparisons the aggregate function-based method, the bitmap index-based method, and the method of using regular SQL queries. They are compared on three different data sets, the own synthetic data, for studying the effect of various parameters in the performance of these methods, including the number of tuples, the number of groups, the number of values in a set predicate, the number of qualified groups, and so on; 2) for studying the performance of these methods on general queries with join conditions and on benchmark data capturing the characteristics of decision support applications; 3) World- Cup98 data set (Section 9.2.3), for evaluating the performance of the methods on real and big data. The aggregate function-based method, denoted as Agg, is implemented in C++. The bitmap index-based method, denoted as Bitmap, is also implemented in C++ and leverages FastBit4 for BSI implementation. The method of using regular SQL to express set-level comparisons is denoted as Rewrite. PostgreSQL is used to store data and execute regular SQL queries. In the supplemental materials, available online, to this paper, it describes how to rewrite queries with set predicate into regular SQL. It is not a complete enumeration of all possible query rewritings because in practice there will be infinite possible rewritings. This was done by manually investigating alternative queries and plans and turning on/off various physical query operators. Nevertheless, the queries we often used for CONTAINED BY are in the form of the rewriting in Fig. 2. For a CONTAIN predicate with m values, we often used a query that intersects the results of m selection queries on the individual values. This rewriting approach can be found in the supplemental materials, available online, to this paper. Moreover, the query plans resulting from regular SQL queries discussed in Section 4 ultimately perform one-pass grouping and aggregation upon the results of (multiple) other upstream operations. Therefore the performance of Agg, which is also implemented externally, serves as a yardstick in comparison with the performance of Bitmap. Hence, the results verify that using regular SQL queries obscures the semantics of set-level comparisons and leads to costly plans. The results could encourage vendors to incorporate the proposed approaches into a database engine.


Add Course Details

Company Name:	Microsoft
CRN:	43212
IDE:	VS
Data Base:	Oracle
Server:	IIS
Code Master:	C#
Basic Concept:	☐oops Concept ☐Operator ☑Array ☑Function ☐Pointer ☐File Management ☐CRUD Queries ☐Trigger ☐Joins ☐Aggregation Function
Advance Concept:	☐Event Handling ☐Thread Animation ☐Delegate ☐DOM parser

Fig.2.CourseAddition

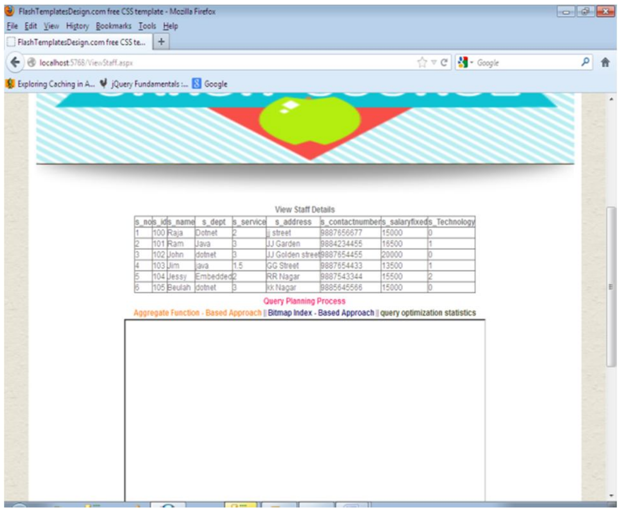


View Feed Back

Email Id	Feed Back
jim@gmail.com	useful
jim@gmail.com	better

Copyright 2014.

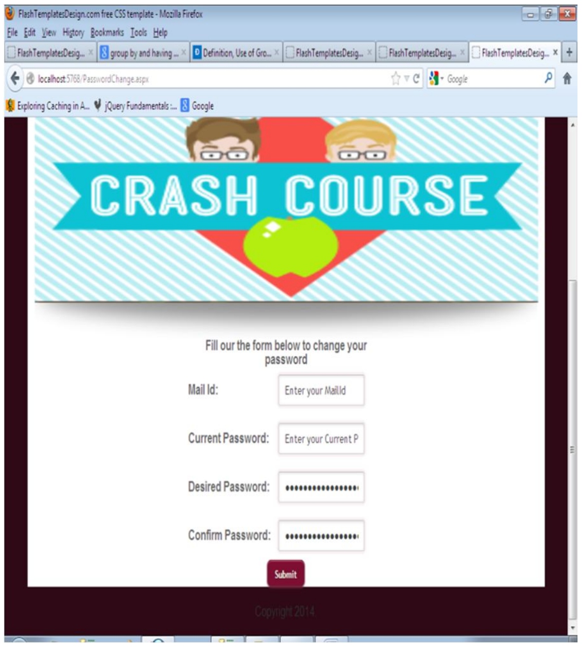
Fig.3. Course Registration


View Staff Details

s_no	s_id	s_name	s_dept	s_service	s_address	s_contactnumber	s_salaryfixed	s_Technology
1	100	Raja	Dotnet	2	jj street	9887656677	15000	0
2	101	Ram	Java	3	JJ Garden	9884234455	16500	1
3	102	John	dotnet	3	JJ Golden street	9887654455	20000	0
4	103	Jim	java	1.5	GG Street	9887654433	13500	1
5	104	Jessy	Embedded	2	RR Nagar	9887543344	15500	2
6	105	Beulah	dotnet	3	kk Nagar	9885645566	15000	0

Query Planning Process
Aggregate Function - Based Approach || Bitmap Index - Based Approach || query optimization statistics

Fig.4. Viewing Staff Details


FlashTemplatesDesign.com free CSS template - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost:5768/PasswordChange.aspx
CRASH COURSE
Submit
Copyright 2014

Fill our the form below to change your password

Mail Id: Enter your MailId

Current Password: Enter your Current P

Desired Password: ••••••••••••••••

Confirm Password: ••••••••••••••••

Fig.5 Changing password for login

VI. Conclusion

It is proposed to extend SQL by set predicates to support set-level comparisons. Such predicates, combined with grouping, allow selection of dynamically formed groups by comparison between a group and a set of values. The two evaluation methods are presented to process set predicates. Comprehensive experiments on synthetic and TPC- H data show the effectiveness of both the aggregate function-based approach and the bitmap index-based approach. For optimizing multi predicate queries, a histogram-based probabilistic method iscreated to estimate the selectivity of set predicates. The estimation governs the evaluation order of multiple predicates, producing efficient query plans

VII. References

1. "Jaql: Query Language for Javascript Object Notation (Json),"http://code.google.com/p/jaql/, 2013.
2. G. Antoshenkov, "Byte-Aligned Bitmap Compression," Proc. Conf. Data Compression, 1995.
3. M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," IEEE Network, vol. 14, no. 3, pp. 30-37, May/June 2000.
4. Chamberlin, M. Astrahan, K. Eswaran, P. Griffiths, R. Lorie, J. Mehl, P. Reisner, and B. Wade, "Sequel 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM J. R & D, vol. 20, no. 6, pp. 560-575, 1976.
5. C.Y. Chan and Y.E. Ioannidis, "An Efficient Bitmap Encoding Scheme for Selection Queries," Proc. ACM SIGMOD Int'l Conf. Management of Data, 1999.
6. A.K. Chandra and P.M. Merlin, "Optimal Implementation of Conjunctive Queries in Relational Data Bases," Proc. Ninth Ann. ACM Symp. Theory of Computing (STOC), 1977.
7. Chatziantoniou, "Using Grouping Variables to Express Com- plex Decision Support Queries," Data Knowledge Eng., vol. 61, no. 1, pp. 114-136, 2007.D. Chatziantoniou and K.A. Ross, "Querying Multiple Features of Groups in Relational Databases," Proc. Int'l Conf. Very Large Databases (VLDB), pp. 295-306, 1996.
8. Chatziantoniou and K.A. Ross, "Groupwise Processing of Relational Queries," Proc. 23rd Int'l Conf. Very Large Databases (VLDB), pp. 476-485, 1997.
9. D. Chatziantoniou and E. Tzortzakakis, "Asset Queries: A Declarative Alternative to Mapreduce," ACM SIGMOD Record, vol. 38, no. 2, pp. 35-41, Oct. 2009.
10. R. Elmasri and S. Navathe, Fundamentals of Database Systems. Addison-Wesley, 2011.

11. Graefe and R.L. Cole, "Fast Algorithms for Universal Quantification in Large Databases," ACM Trans. Database Systems, vol. 20, no. 2, pp. 187-236, 1995
12. [12] J.M. Hellerstein and M. Stonebraker, "Predicate Migration: Optimizing Queries with Expensive Predicates," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 267-276, 1993.
13. S. Helmer and G. Moerkotte, "Evaluation of Main Memory Join Algorithms for Joins with Set Comparison Join Predicates," Proc. Int'l Conf. Very Large Databases (VLDB), 1996.
14. Y. Ioannidis, "The History of Histograms (Abridged)," Proc. Int'l Conf. Very Large Databases (VLDB), 2003.