International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
Volume 2 Issue 1 –OCTOBER 2013.

An Improving Performance of Filter Method Using Feature Subset Selection Algorithm

L. Gomathi[#1], S. Kanimozhi[*2]

[#]*Associate Professor, Dept. of Computer Science, Muthayammal College of Arts & Science, TN, India*
[2] kanics10@gmail.com
[#] *Dept. of Computer Science, Muthayammal College of Arts & Science, TN, India*

Abstract: **To analyze data efficiently, Data mining systems are widely using datasets with columns in horizontal tabular layout. Preparing a data set is more complex task in a data mining project, requires many SQL queries, joining tables and aggregating columns. Conventional RDBMS usually manage tables with vertical form. Aggregated columns in a horizontal tabular layout returns set of numbers, instead of one number per row. The system uses one parent table and different child tables, operations are then performed on the data loaded from multiple tables. PIVOT operator, offered by RDBMS is used to calculate aggregate operations. Relational databases are acceptable repository for structured data; integrating data mining algorithms with a relational DBMS is an essential research issue for database programmers. In a relational database, a significant effort is required to prepare a summary data set that can be used as input for the data mining process.**

Index Terms- Aggregation, Data Mining, Structured query language (SQL), PIVOT, Lattice.

I. INTRODUCTION

Horizontal aggregation is new class of function to return aggregated columns in a horizontal layout. Most algorithms require datasets with horizontal layout as input with several records and one variable or dimensions per columns. Managing large data sets without DBMS support can be a difficult task. Trying different subsets of data points and dimensions is more flexible, faster and easier to do inside a relational database with SQL queries than outside with alternative tool. Horizontal aggregation can be performing by using operator, it can easily be implemented inside a query processor, much like a select, project and join. PIVOT operator on tabular data that exchange rows, enable data transformations useful in data modelling, data analysis, and data presentation. There are many existing functions and operators for aggregation in Structured Query Language. The most commonly used aggregation is the sum of a column and other aggregation operators return the average, maximum, minimum or row count over groups of rows.

All operations for aggregation have many limitations to build large data sets for data mining purposes. Database schemas are also highly normalized for On-Line Transaction Processing (OLTP) systems where data sets that are stored in a relational database or data warehouse. But data mining, statistical or machine learning algorithms generally require aggregated data in summarized form. Data mining algorithm requires suitable input in the form of cross tabular (horizontal) form, significant effort is required to compute aggregations for this purpose. Such effort is due to the amount and complexity of SQL code which needs to be written, optimized and tested. Data aggregation is a process in which information is gathered and expressed in a summary form, and which is used for purposes such as statistical analysis. A common aggregation purpose is to get more information about particular groups based on specific variables such as age, name, phone number, address, profession, or income. Most algorithms require input as a data set with a horizontal layout, with several records and one variable or dimension per column. That technique is used with models like clustering, classification, regression and PCA. Dimension used in data mining technique are point dimension.

II. HORIZONTAL AGGREGATIONS

We introduce a new class of aggregations that have similar behavior to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, we call standard SQL aggregations vertical aggregations since they produce tables with a vertical layout. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement. Alternatively, horizontal aggregations can be used to generate SQL code from a data mining tool to build data sets for data mining analysis. We start by explaining how to automatically generate SQL code.

III. BACKGROUND WORK

To prepare summarized format for data mining algorithm, many methods are introduced by researchers in the past. Carlos Ordonez [3] introduced three SQL implementations of the popular K-means clustering algorithm to integrate it with a relational DBMS. Xiaoxin Yin [14] proposed a new approach, called CrossMine, which includes a set of novel and powerful methods for multirelational classification. Carlos Ordonez [2] focused on programming Bayesian classifiers in SQL. Carrasco [6] defined a new type of object dmFSQL consists of a series of operations on the object project (create, alter, drop…). The DML of dmFSQL executes the true DM process. Elena Baralis[9] presented the IMine indx, a general and compact structure which provides tight integration of item set extraction in a relational DBMS. Charu C. Aggarwal[7] provided a survey of uncertain data mining and management applications. Sally McClean[11] considered the problem of aggregation using an imprecise probability data model. Conor Cunningham [8] described PIVOT and UNPIVOT, two operators on tabular data that exchange rows and columns. Haixun Wang [14] implemented ATLaS, to develop complete data-intensive applications in SQL—by writing new aggregates and table functions in SQL, it includes query rewriting, optimization techniques and the data stream management module.Carlos Ordonez [1] introduced techniques to efficiently compute fundamental statistical models inside a DBMS exploiting User-Defined Functions (UDFs).

Algorithm: (Alignment)

Detection System ti:

collect raw alerts ri locally

// LAi : correlate – and – filter (ri,ti)

Al⇐ correlate and filter(ri,1)

for each pij Є Al do

// look up destination node for pij

dt=lookup(srcIP of pij)

subscribe(pij,nij,dj) on dt

end for

end for

IV. PROPOSED WORK

The main goal is to define a template to generate SQL code by combining aggregation and transposition. The proposal has two perspectives such as to evaluate efficient aggregations and perform query optimization. The first one includes the following approaches, pivoting, transposition and cross-tabulation. Pivoting approach is a built-in method in a commercial DBMS. It can help evaluating an aggregated tabular format for summarized data set. It perform the following steps, The pivoting method is used to write cross-tabulation queries that rotate rows into columns, aggregating data in the process of the rotation. The output of a pivot operation typically includes more columns and fewer rows than the starting data set. The pivot computes the aggregation functions specified at the beginning of the clause. Aggregation functions must specify a GROUP BY clause to return multiple values; the pivot performs an implicit GROUP BY. New columns corresponding to values in the pivot, each aggregated value is transposed to the appropriate new column in the cross-tabulation. The subclauses of the pivot have the following semantics: expr - specify an expression that evaluates to a constant value of a pivot column. Subquery – to specify a subquery, all values found by the subquery are used for pivoting. The subquery must return a list of unique values at the execution time of the pivot query.

V. RESULT ANALYSIS

Comparing Evaluation Methods

On the other hand, the second important issue is automatically generating unique column names. If there are many subgrouping columns R_1; . . .;R_k or columns are of string data types, this may lead to generate very long column names, which may exceed DBMS limits. However, these are not important limitations because if there are many dimensions that is likely to correspond to a sparse matrix (having many zeroes or nulls) on which it will be difficult or impossible to compute a data mining model. On the other hand, the large column name length can be solved as explained below. The problem of d going beyond the maximum number of columns can be solved by vertically partitioning FH so that each partition table does not exceed the maximum number of columns allowed by the DBMS. Evidently, each partition table must have L_1; . . . ; L_j as its primary key. Alternatively, the column name length issue can be solved by generating column identifiers with integers and creating a “dimension” description table that maps identifiers to full descriptions, but the meaning of each dimension is lost. An alternative is the use of abbreviations, which may require manual input.

Query Optimizations

Our first query optimization, applied to three methods. Our goal is to assess the acceleration obtained by precomputing a cube and storing it on F_V . We can see this optimization uniformly accelerates all methods. This optimization provides a different gain, depending on the method: for SPJ the optimization is best for small n, for PIVOT for large n and for CASE there is rather a less dramatic improvement all across n. It is noteworthy PIVOT is accelerated by our optimization, despite the fact it is handled by the query optimizer. Since this optimization produces significant acceleration for the three methods (at least 2 faster) we will use it by default. Notice that precomputing F_V takes the same time within each method. Therefore, comparisons are fair. We now evaluate an optimization specific to the PIVOT operator. This PIVOT optimization is well known, as we learned from SQL Server DBMS users groups. shows the impact of removing (trimming) columns not needed by PIVOT. That is, removing columns that will not appear in F_H. We can see the impact is significant, accelerating evaluation time from three to five times. All our experiments incorporate this optimization by default.

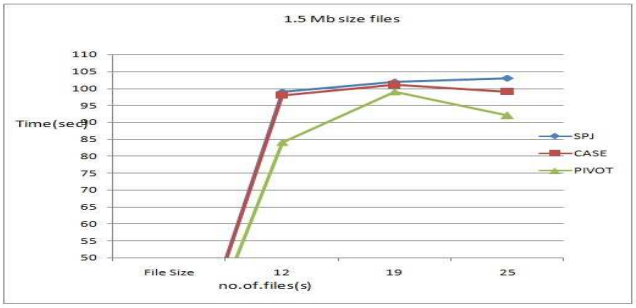

Figure 1: Time complexity varying d (1.5 mb, uniform distribution).

Time Complexity

We now verify the time complexity analysis given in We plot time complexity keeping varying one parameter and the remaining parameters fixed. In these experiments, we generated synthetic data sets similar to the fact table of TPC-H of different sizes with grouping columns of varying selectivities (number of distinct values). We consider two basic probabilistic distribution of values: uniform (unskewed) and zipf (skewed). The uniform distribution is the distribution used by default.

VI. CONCLUSION

The proposed approaches implements an abstract but minimal extension to SQL standard aggregate functions to compute efficient summarized data set which just requires specifying sub grouping columns inside the aggregation function call. From a query optimization perspective, The proposed system describes the possibility of extending SQL OLAP aggregations with horizontal layout capabilities. Horizontal aggregations produce tables with fewer rows, but with more columns. The aggregated tables are useful to create data sets with a horizontal layout, as commonly required by data mining algorithms and OLAP cross-tabulation. The output of a query optimization can immediately be applied back to the data gathering, transformation, and analysis processes. Anomalous data can be detected in existing data sets, and new data entry can be validated in real time, based on the existing data. SQL Server Data Mining contains multiple algorithms that can perform churn analysis based on historical data. Each of these algorithms will provide a probability. In future, research issues is proposed on extending SQL code for data mining processing. The related work on query optimization is proposed and compared to horizontal aggregations with alternative proposals to perform transposition or pivoting. It includes to develop more complete I/O cost models for cost-based query optimization and to study optimization of horizontal aggregations processed in parallel in a shared-nothing DBMS architecture.

VII. REFERENCES

[1] [1] Carlos Ordonez,‖ Statistical model computation with UDFs‖, IEEE Transactions on Knowledge and Data Engineering, vol. 22, no.22, pp. 1752 - 1765, Dec. 2010.

[2] [2] Carlos Ordonez, Pitchaimalai. S.K, ―Bayesian Classifiers Programmed in SQL‖, IEEE Trans. Knowledge and Data Eng, vol. 22, no. 1, pp.909-921, Jan. 2010.

[3] [3] Carlos Ordonez, ―Integrating K-means clustering with a relational DBMS using SQL‖ IEEE Trans. Knowledge and Data Eng, vol. 18 no. 2, pp.181-201, Feb. 2006

[4] [4] Carlos Ordonez, Omiecinski. E, ―Efficient Disk-Based K-Means Clustering for Relational Databases‖, IEEE Trans. Knowledge and Data Eng., vol. 16, no. 8, pp.909-921, Aug. 2004.

[5] [5] Carlos Ordonez, Zhibo Chen, ―Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis‖, IEEE Trans. Knowledge and Data Eng., vol. PP, no. 99, Jan. 2011.

[6] [6] Carrasco, R.A.; Vila, M.A.; Araque, F.,‖ dmFSQL: a Language for Data Mining‖, DEXA '06. 17th International Workshop on 2006, pp-440-444, 2006

[7] [7] Charu C. Aggarwal, Philip S. Yu. ―A Survey of Uncertain Data Algorithms and Applications‖, IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 5. pp. 609-623, May 2009.

[8] [8] Cunningham.C, Graefe.G, and Galindo-Legaria.C.A, PIVOT and UNPIVOT: Optimization and execution strategies in an RDBMS, In Proc. VLDB Conference, pages 998–1009, 2004.

[9] [9] Elena Baralis, Tania Cerquitelli, Silvia Chiusano, "IMine: Index Support for Item Set Mining," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no.4, pp 493-506, April 2009