ISSN:0975-9646

Implementing an Efficient Task to Build Data Sets for Datamining Analysis.

B.K.Manasa[1], H.Venkateswara Reddy[2]
*M.Tech-SE, Dept.,Of CSE., Vardhaman College Of Engg, Hyderabad ,India
**Professor & Head Of Dept.,CSE.,Vardhaman College Of Engg,Hyderabad ,India

Abstract- Data mining is the process of discovering actionable information from large sets of data. Preparing a data set for analysis is generally the most time consuming task in a data mining project, requiring many complex sql queries, joining tables and aggregating columns. Existing sql aggregations have limitations to prepare data sets because they return one column per aggregated group. This paper presents Methodology of Horizontal aggregations using a method to generate sql code to return aggregated columns in a horizontal tabular layout, returning a set of numbers instead of one number per row. There are three fundamental variations to evaluate horizontal aggregations. CASE: Exploiting the programming CASE construct; SPJ: Based on standard relational algebra operators (SPJ queries); PIVOT: Using the PIVOT operator, which is offered by some DBMSs. The proposed methodology shows evaluating horizontal aggregations is a challenging and interesting problem and introduces alternative methods and optimizations for their efficient evaluation.

Index terms: aggregation; data preparation; pivoting; SQL

I. INTRODUCTION

In data mining the common terms are point-dimension. Statistics literature generally uses observation-variable. Machine learning research uses instance-feature. Most algorithms require as input a data set with a horizontal layout, with several records and one variable or dimension per column. That is the case with models like clustering, classification, regression and PCA.

A. Motivation

As mentioned above, building a suitable data set for data mining purposes is a time-consuming task. This task generally requires writing long SQL statements or customizing SQL code if it is automatically generated by some tool. There are two main ingredients in such SQL code: joins and aggregations we focus on the second one. The most widely-known aggregation is the sum of a column over groups of rows. Some other aggregations return the average, maximum, minimum or row count over groups of rows. There exist many aggregation functions and operators in SQL. Unfortunately, all these aggregations have limitations to build data sets for data mining purposes. The main reason is that, in general, data sets that are stored in a relational database (or a data warehouse) come from On-Line Transaction Processing (OLTP) systems where database schemas are highly normalized. But data mining, statistical or machine learning algorithms generally require aggregated data in summarized form. Based on current available functions and clauses in SQL, a significant effort is required to compute aggregations when they are desired in a crosstabular (horizontal) form, suitable to be used by a data mining algorithm. Such effort is due to the amount and complexity of SQL code that needs to be written, optimized and tested.. Standard aggregations are hard to interpret when there are many result rows, especially when grouping attributes have high cardinalities. To perform analysis of exported tables into spreadsheets it may be more convenient to have aggregations on the same group in one row (e.g. to produce graphs or to compare data sets with repetitive information). OLAP tools generate SQL code to transpose results (sometimes called PIVOT).Transposition can be more efficient if there are mechanisms combining aggregation and transposition together. With such limitations in mind, we propose a new class of aggregate functions that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout. Functions belonging to this class are called horizontal aggregations. Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout (somewhat similar to a multidimensional vector), instead of a single value per row.

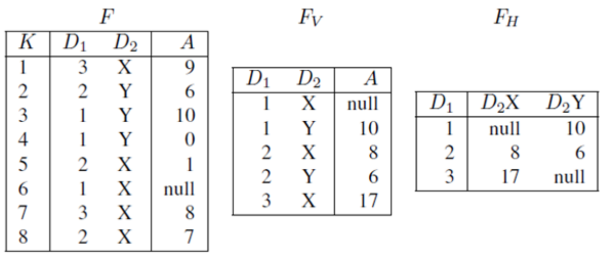
B.Typical Data Mining Problems

Let us consider data mining problems that may be solved by typical data mining or statistical algorithms, which assume each non-key column represents a dimension, variable (statistics) or feature (machine learning). Stores can be clustered based on sales for each day of the week. On the other hand, we can predict sales per store department based on the sales in other departments using decision trees or regression. PCA analysis on department sales can reveal which departments tend to sell together. We can find out potential correlation of number of employees by gender within each department. Most data mining algorithms (e.g. clustering, decision trees, regression, correlation analysis) require result tables from these queries to be transformed into a horizontal layout. We must mention there exist data mining algorithms that can directly analyze data sets having a vertical layout (e.g. in transaction format) [14], but they require reprogramming the algorithm to have a better I/O pattern and they are efficient only when there many zero values (i.e. sparse matrices).

The rest of the paper is organized as follows: Section II gives a glance on Horizontal Aggregations. Section III describes the recent research. Section IV describes the proposed process work. Section V illustrates the experimental evaluation and finally Section VI concludes the paper.

II .HORIZONTAL AGGREGATIONS

The 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.



F

K	D_1	D_2	A
1	3	X	9
2	2	Y	6
3	1	Y	10
4	1	Y	0
5	2	X	1
6	1	X	null
7	3	X	8
8	2	X	7

F_V

D_1	D_2	A
1	X	null
1	Y	10
2	X	8
2	Y	6
3	X	17

F_H

D_1	D_2X	D_2Y
1	null	10
2	8	6
3	17	null

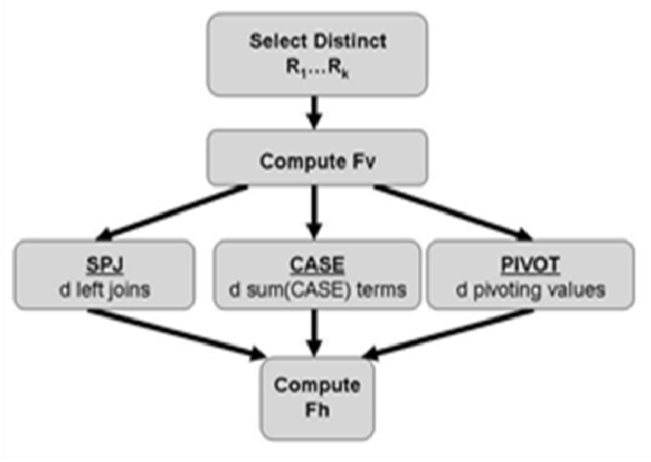
Fig. 1 – Input table (a), traditional vertical aggregation (b), and horizontal aggregation (c)

As can be seen in fig. 1, input table has some sample data. Traditional vertical sum aggregations are presented in (b) which is the result of SQL SUM function while (c) holds the horizontal aggregation which is the result of SUM function.

III. RECENT RESEARCH

First discuss research on extending SQL code for data mining processing. We briefly discuss related work on query optimization. We then compare horizontal aggregations with alternative proposals to perform transposition or pivoting. There exist many proposals that have extended SQL syntax. The closest data mining problem associated to OLAP processing is association rule mining . SQL extensions to define aggregate functions for association rule mining are introduced. In this case the goal is to efficiently compute item set support. Unfortunately, there is no notion of transposing results since transactions are given in a vertical layout. Programming a clustering algorithm with SQL queries is explored in , which shows a horizontal layout of the data set enables easier and simpler SQL queries. The PIVOT and CASE methods avoid joins as well. Our SPJ method proved horizontal aggregations can be evaluated with relational algebra, exploiting outer joins, showing our work is connected to traditional query optimization. Traditional query optimizers use a tree based execution plan, but there is work that advocates the use of hyper-graphs to provide a more comprehensive set of potential plans. This approach is related to our SPJ method. Even though the CASE construct is an SQL feature commonly used in practice optimizing queries that have a list of similar CASE statements has not been studied in depth before. Research on efficiently evaluating queries with aggregations is extensive. We focus on discussing approaches that allow transposition, pivoting or cross-tabulation. The importance of producing an aggregation table with a cross-tabulation of aggregated values is recognized in the context of cube computations. An operator to unpivot a table producing several rows in a vertical layout for each input row to compute decision trees was proposed. Several SQL primitive operators for transforming data sets for data mining were introduced ; the most similar one to ours is an operator to transpose a table, based on one chosen column. The TRANSPOSE operator is equivalent to the unpivot operator, producing several rows for one input row. An important difference is that, compared to PIVOT, TRANSPOSE allows two or more columns to be transposed in the same query, reducing the number of table scans. Therefore, both UNPIVOT and TRANSPOSE are inverse operators with respect to horizontal aggregations. Later, SQL operators to pivot and unpivot a column were introduced (now part of the SQL Server DBMS); this work took a step beyond by considering both complementary operations: one to transpose rows into columns and the other one to convert columns into rows (i.e. the inverse operation). There are several important differences with our proposal though: the list of distinct to values must be provided by the user, whereas ours does it automatically; output columns are automatically created; the PIVOT operator can only transpose by one column, whereas ours can do it with several columns, the PIVOT operator requires removing unneeded columns (trimming) from the input table for efficient evaluation (a well-known optimization to users), whereas ours an work directly on the input table. Horizontal aggregations are related to horizontal percentage aggregations. Finally, our present article is a significant extension of the preliminary work presented, where horizontal aggregations were first proposed.

IV.PROPOSED PROCESS WORK



1 SPJ Method

This method is based on the relational operators only. In this method one table is created with vertical aggregation for each column.

Then all such tables are joined in order to generate a table containing horizontal aggregations.

2. PIVOT Method

This aggregation is based on the PIVOT operator available in RDBMS. As it can provide transpositions, it can be used to evaluate horizontal aggregations. PIVOT operator determines how many columns are required to hold transpose and it can be combined with GROUP BY clause.

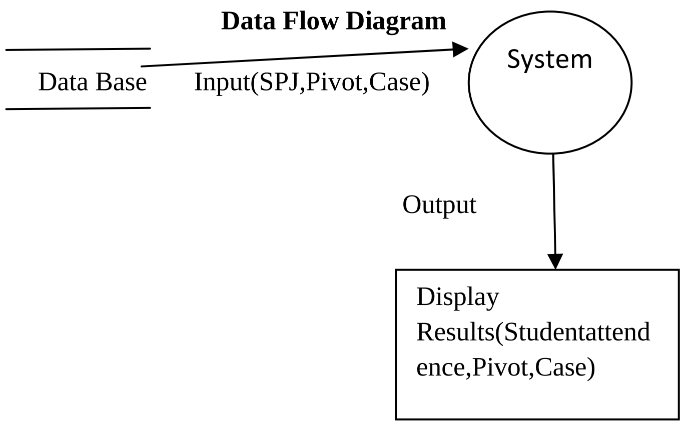
3. CASE Method

This construct is built based on the existing CASE construct of SQL. Based on Boolean expression one of the results is returned by CASE construct.

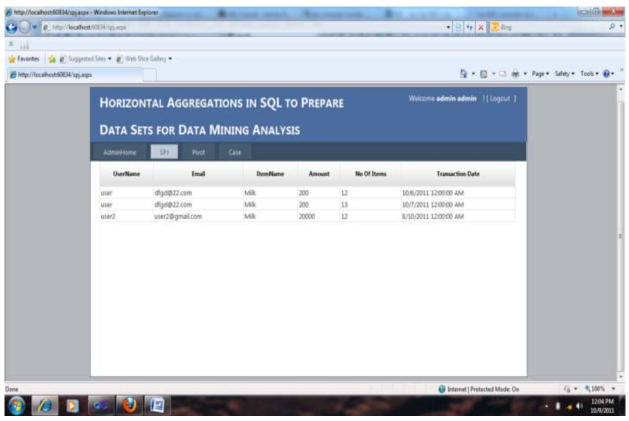
It is same as projection/aggregation query from relational point of view. Based on some conjunction of conditions each non key value is given by a function. Here two basic strategies to compute horizontal aggregations. The first strategy is to compute directly from input table. The second approach is to compute vertical aggregation and save the results into temporary table. Then that table is further used to compute horizontal aggregations.

V.EXPERIMENTAL EVALUATION

The environment used for the development of prototype web based application that demonstrates the three horizontal aggregations include Visual Studio 2010, and SQL Server 2008. The former is used to build front end application with web based interface while the latter is used to store data permanently. The technologies used include ASP.NET which is meant for developing web services and web applications, and AJAX (Asynchronous JavaScript and XML) for rich user experience.



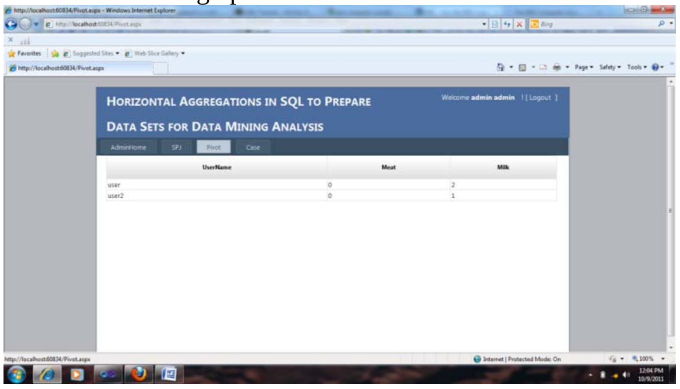
Programming language used in C# which is an object oriented high level programming language. The result of horizontal aggregation SPJ is shown in fig


Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis
Welcome admin admin | [Logout]
AdminHome
SPJ
Pivot
Case

UserName	Email	ItemName	Amount	No Of Items	Transaction Date
user	dfgd@22.com	Milk	200	12	10/6/2011 12:00:00 AM
user	dfgd@22.com	Milk	200	13	10/7/2011 12:00:00 AM
user2	user2@gmail.com	Milk	20000	12	8/10/2011 12:00:00 AM

Fig. 1 – Results of SJP aggregation

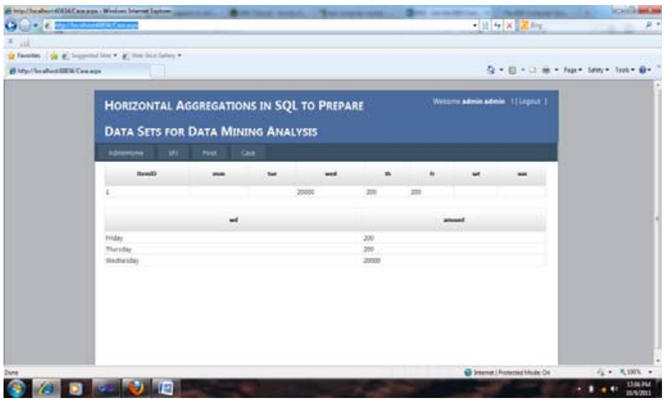
As can be seen in fig. 1, the results are through the SPJ operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.


Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis
Welcome admin admin | [Logout]
AdminHome
SPJ
Pivot
Case

UserName	Meat	Milk
user	0	2
user2	0	1

Fig. 2 – Result of Pivoting Aggregation

As can be seen in fig. 2, the results are through the PIVOT operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.


Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis
Welcome admin admin | [Logout]
AdminHome
SPJ
Pivot
Case

ItemID	mon	tue	wed	th	fr	sat	sun
1			20000	200	200		

wd	amount
Friday	200
Thursday	200
Wednesday	20000

Fig. 3 – Result of CASE Aggregation

As can be seen in fig. 2, the results are through the CASE operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.

VI.CONCLUSION

In this paper, we presented a methodology of Horizontal aggregations using a method to generate sql code to return aggregated columns in a horizontal tabular layout, returning a set of numbers instead of one number per row. It show evaluating horizontal aggregations is a challenging and interesting problem and introduced alternative methods and optimizations for their efficient evaluation. Introduced a new class of extended aggregate functions, called horizontal aggregations which help preparing data sets for data mining and OLAP cube exploration. From a query optimization perspective, we proposed three query evaluation methods. The first one (SPJ) relies on standard relational operators. The second one (CASE) relies on the SQL CASE construct. The third (PIVOT) uses a built-in operator in a commercial DBMS that is not widely available. The proposed horizontal aggregations can be used as a database method to automatically generate efficient SQL queries with three sets of parameters: grouping columns, subgrouping columns and aggregated column. Efficiently evaluating horizontal aggregations using left outer joins presents opportunities for query optimization. Secondary indexes on common grouping columns, besides indexes on primary keys, can accelerate computation.The proposed horizontal aggregations do not introduce conflicts with vertical aggregations, but we need to develop a more formal model of evaluation.

REFERENCES

[1] G. Bhargava, P. Goel, and B.R. Iyer. Hypergraph based reorderings of outer join queries with complex predicates. In ACM SIGMOD Conference, pages 304–315, 1995.

[2] J.A. Blakeley, V. Rao, I. Kunen, A. Prout, M. Henaire, and C. Kleinerman. .NET database programmability and extensibility in Microsoft SQL Server. In Proc. ACM SIGMOD Conference, pages 1087–1098, 2008.

[3] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman. Non-stop SQL/MX primitives for knowledge discovery. In ACM KDD Conference, pages 425–429, 1999.

[4] E.F. Codd. Extending the database relational model to capture more meaning. ACM TODS, 4(4):397–434, 1979.

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

[6] C. Galindo-Legaria and A. Rosenthal. Outer join simplification and reordering for query optimization. ACM TODS, 22(1):43–73, 1997.

[7] H. Garcia-Molina, J.D. Ullman, and J. Widom. Database Systems: The Complete Book. Prentice Hall, 1st edition, 2001.

[8] G. Graefe, U. Fayyad, and S. Chaudhuri. On the efficient gathering of sufficient statistics for classification from large SQL databases. In Proc. ACM KDD Conference, pages 204–208, 1998.

[9] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub- total. In ICDE Conference, pages 152–159, 1996.

[10] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco, 1st edition, 2001.

[11] G. Luo, J.F. Naughton, C.J. Ellmann, and M. Watzke. Locking protocols for materialized aggregate join views. IEEE Transactions on Knowledge and Data Engineering (TKDE), 17(6):796–807, 2005.

[12] C. Ordonez. Horizontal aggregations for building tabular data sets. In Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop, pages 35–42, 2004.