ABSTRACT: In Data mining, an essential objective is to create efficient data. Effectiveness and scalability have always been important concern in the field of data mining. The increased complexity of the task calls for algorithms that are inherently more expensive. To examine 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. PIVOT method is much faster method and offers much scalability. Partitioning large set of data, obtained from the result of horizontal aggregation, in to homogeneous cluster is important task in this system. K-means algorithm using SQL is best suited for implementing this operation.

KEYWORDS: aggregation; data preparation; pivoting; SQL

INTRODUCTION

In a relational database, especially with normalized tables, a significant effort is required to prepare a summary data set that can be used as input for a data mining or statistical algorithm. Most algorithms require as input a data set with a horizontal layout, with several records and one variable or dimension per column. Consider the case with models like clustering, classification, regression, and PCA; each research discipline uses different terminology to describe the data set. In data mining the common terms are point-dimension. Statistics literature generally uses observation-variable. Machine learning research uses instance feature. This report introduces a new class of aggregate functions that can be used to build data sets in a horizontal layout (demoralized with aggregations), automating SQL query writing and extending SQL capabilities. A show evaluating horizontal aggregations is a challenging and interesting problem and A introduces alternative methods and optimizations for their efficient evaluation. 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 aggregations 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 analysis, and data presentation. There are many existing functions and operators for aggregation in SQL. The most commonly used aggregations is the sum of a column (instance aggregations), automating SQL query writing and extending SQL capabilities.
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. MOTIVATION

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. 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 Online 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 cross tabular (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. There are further practical reasons to return aggregation results in a horizontal (cross-tabular) layout. Standard aggregations are hard to interpret when there are many result rows, especially when grouping attributes have high cardinalities.

Motivation examples:

Consider the analysis of sales details in various stores. Suppose there are 50 stores and stores are open 7 days a week. Then the above query will give 350 rows. For these type of analysis the data mining algorithms should require storied as primary key and the remaining columns as non-key. It means that the data mining algorithms require the result in horizontal layout. Also, the data set in horizontal format can be analysed more efficient than data set in vertical format.
### T

<table>
<thead>
<tr>
<th>Storeid</th>
<th>Salesday</th>
<th>Salesamt</th>
<th>Billno</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Monday</td>
<td>1500</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>Tuesday</td>
<td>2500</td>
<td>102</td>
</tr>
<tr>
<td>2</td>
<td>Monday</td>
<td>1800</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>Tuesday</td>
<td>1200</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>Tuesday</td>
<td>1400</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>Tuesday</td>
<td>1500</td>
<td>102</td>
</tr>
<tr>
<td>1</td>
<td>Monday</td>
<td>1200</td>
<td>103</td>
</tr>
<tr>
<td>2</td>
<td>Tuesday</td>
<td>1800</td>
<td>103</td>
</tr>
<tr>
<td>3</td>
<td>Wednesday</td>
<td>2000</td>
<td>103</td>
</tr>
</tbody>
</table>

### T^T

<table>
<thead>
<tr>
<th>Storeid</th>
<th>Salesday</th>
<th>Salesamt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Monday</td>
<td>2700</td>
</tr>
<tr>
<td>1</td>
<td>Tuesday</td>
<td>2500</td>
</tr>
<tr>
<td>2</td>
<td>Monday</td>
<td>1800</td>
</tr>
<tr>
<td>2</td>
<td>Tuesday</td>
<td>3000</td>
</tr>
<tr>
<td>3</td>
<td>Tuesday</td>
<td>2900</td>
</tr>
<tr>
<td>3</td>
<td>Wednesday</td>
<td>2000</td>
</tr>
</tbody>
</table>

### T^H

<table>
<thead>
<tr>
<th>Storeid</th>
<th>Salesman</th>
<th>Salestue</th>
<th>Saleswed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2700</td>
<td>2500</td>
<td>Null</td>
</tr>
<tr>
<td>2</td>
<td>1800</td>
<td>3000</td>
<td>Null</td>
</tr>
<tr>
<td>3</td>
<td>Null</td>
<td>2900</td>
<td>2000</td>
</tr>
</tbody>
</table>
III. OBJECTIVES

1) The proposed dissertation objective is a new class of extended aggregate functions, called horizontal aggregations which help in investigating data sets for data mining and OLAP cube exploration.

2) It also investigate three query evaluation method for horizontal aggregation in order to optimize the query required for preparing summary data sets and the three methods are SPJ, PIVOT and CASE.

3) Combining any of three methods (SPJ, PIVOT, and CASE) with horizontal aggregations used to generate SQL code to build data set for data mining. Aim is to provide a more efficient, better integrated and more secure solution compared to external data mining tools.

A. SPJ METHOD

The SPJ method is based on the relational operators such as select, join and project. The idea is to create the aggregated result for each values of the attributes that to be transposed and the result is stored in individual tables say Ti (i=1,……p),for each distinct value of m1 ,m2 …….mv . These tables are then left outer joined with the table R0 ,that contain the distinct values of n1 ,n2 ……nt . Left outer join is used to deal with missing information. This operation is performed by taking all tuples in the left relation that did not match with any tuple in the right relation and set the attributes from the right relation to null. Horizontal aggregation queries can be evaluated by aggregating data directly from R based on the distinct values of grouping attributes. So the distinct values of m1,m2,……,mv are to identified, that define the matching Boolean expression for result columns. These results are then transposed to produce RH. Agg() is the standard vertical aggregation that has aggregate attribute as argument. The SQL code for generating R0 :

```sql
SELECT DISTINCT n1 ,n2 ,….,nt
FROM R
```

Tables T1 ,T2 .....Tl contain the individual aggregations for each combination of m1 ,m2 ,……,mV. { n1 ,n2 ,….,nt } become the key of RI .

SQL code for generating RI :

```sql
SELECT n1 ,n2 ,….,nt , Agg(X)
FROM T
WHERE m1=v1I
AND…..AND mV
GROUP BY n1 ,n2 ,….,nt
```

The final SQL code for SPJ method

```sql
SELECT T0 .n1 ,T0 .n2 ,……,T0 .nt , T1 .X , T2 .X ,……, T1 .X FROM T0
LEFT OUTER JOIN T1 ON T0 .n1 = T1 .n1 and…..and T0 .nt = T1 .nt
LEFT OUTER JOIN R2 ON T0 .n1 = T2 .n1 and…..and T0 .nt = T2 .nt ...
LEFT OUTER JOIN Tl
ON T0 .n1 = Td .n1 and…..and T0 .nt = Tl .nt
```

Main steps for SPJ method
B. CASE METHOD

For this method, use the “case” programming construct available in SQL. The case statement returns a value selected from a set of values based on boolean expressions. From a relational database theory point of view this is equivalent to doing a simple projection/aggregation query where each nonkey value is given by a function that returns a number based on some conjunction of conditions. The two basic substrategies to compute FH. In a similar manner to SPJ, the first one directly aggregates from F and the second one computes the vertical aggregation in a temporary table FV and then horizontal aggregations are indirectly computed from FV. A now present the direct aggregation method. Horizontal aggregation queries can be evaluated by directly aggregating from F and transposing rows at the same time to produce FH. First, there is need to get the unique combinations of R1, . . ., Rk that define the matching boolean expression for result columns. The SQL code to compute horizontal aggregations directly from F is as follows: V () is a standard (vertical) SQL aggregation that has a “case” statement as argument. Horizontal aggregations need to set the result to null when there are no qualifying rows for the specific horizontal group to be consistent with the SPJ method and also with the extended relational model [4]. The CASE method code is as follows (computed from F):

```
INSERT INTO FH SELECT D1,
SUM(CASE WHEN D2='X' THEN A ELSE null END) as D2_X,
SUM(CASE WHEN D2='Y' THEN A ELSE null END) as D2_Y
FROM F
GROUP BY D1;
```

IV. CONCLUSION

A novel category of aggregate functions in SQL called horizontal aggregation is used that helps for construction datasets for the data mining projects. These aggregate functions are used for creating data sets in horizontal layout, because most of the data mining algorithms entail datasets in horizontal layout. Mainly, the presented SQL aggregations revisit results in one column per aggregated group. But in horizontal aggregation, it returns a set of numbers instead of a single numeral for each group. Three query evaluation methods are proposed. The first method
focuses on the relational operators in SQL. The second method focuses on the SQL case construct. The third method focuses on the PIVOT built-in operator in a profitable DBMS. These three methods are then analysed using the large dataset and observe the time and space complexity of each method. By analysing, it is observed that the CASE method is the most efficient method. Horizontal aggregation produces tables with fewer rows but with more columns. So the conventional query optimization techniques are inappropriate for the new class of aggregations. So the next plan is to develop the most proper query optimization procedure for the horizontal aggregation.

REFERENCES

[14] “Vertical and Horizontal Percentage Aggregations”, Carlos Ordonez Teradata, NCR San Diego, CA 92127, USA.