IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

nCORETech 2017

LBS College of Engineering, Kasaragod

Vol. 6, Special Issue 3, March 2017

Horizontal Aggregtions in SQL to Prepare Datasets for Data Mining Analysis

Anjali N.V.

M.Tech, Computer Science Dept, L.B.S. College of Engineering, Kasaragod, India

Abstract: Preparing a data set for analysis is 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, and a significant manual effort is required to build data sets. In this paper proposing simple, efficient methods make SQL code return multiple columns in horizontal aggregation tables. It will return set of values instead of one value for one aggregation query. These functions of class are called as horizontal aggregations. It build data sets with a horizontal de normalized layout, which is the standard layout required by most data mining algorithms. In order to transform the data into suitable form three fundamental horizontal aggregation methods are used: SPJ (select, project, and join) method, CASE method and PIVOT method. The analysis become more efficient if the dataset obtained is in the horizontal form.

Index Terms: SQL Operators, Aggregate functions, Data Set Preparation.

I. INTRODUCTION

In a relational database environment with normalized tables, a significant effort is required to prepare a summary dataset in order to use it as input for a data mining algorithm. Generally collecting the information from databases for analysis is time taking and complex. In data mining projects analysis of data requires complex queries, aggregations, joining tables, maintaining primary and foreign keys. These make data analysis typical and time consuming. Most algorithms from data mining, statistics and machine learning require a dataset to be in horizontal tabular form. In general, external effort is kept on creation of datasets at the time of horizontal layout is required. This article introduces a new class of aggregate functions that can be used to build datasets in a horizontal layout, automating SQL query writing and extending SQL capabilities. 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. There exist many aggregation functions and operators in SQL. Unfortunately, all these aggregations have limitations to build datasets for data mining purposes. The main reason is that, in general, datasets 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. 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 spread sheets it may be more convenient to have aggregations on the same group in one row (e.g. to produce graphs or to compare datasets with repetitive information). OLAP tools generate SQL code to transpose i.e., PIVOT results. Transposition can be more efficient if there are mechanisms combining aggregation and transposition together. With such limitations in mind, propose a new class of aggregate functions that aggregate numeric expressions and transpose results to produce a dataset 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. This article explains how to evaluate and optimize horizontal aggregations generating standard SQL code.

II. AGGREGATION

Horizontal aggregations propose a new class of functions that aggregate numeric expressions and the result are transposed to produce data sets with a horizontal layout. The operation is needed in a number of data mining tasks, such as unsupervised classification and data summation, as well as segmentation of large mixed datasets into smaller uniform subsets that can be easily manage, separately model and analyzed. To create datasets for data mining related works, efficient and summary of data are needed. Database as their nature contains large amount of data. To extract information from the database Structured Query Languages are used. SQL commonly used for the aggregation of large volumes of data. With the help of aggregation details in one table can be aggregated with details in another table. Aggregation functions play a major in the summarization of tables. Normal SQL aggregation functions are sum ( ), avg ( ), min ( ), max ( ) and count ( ).

A. Dataset Preparation

Dataset preparation is very important for any of the operations in the data mining analysis. Preparation of dataset addresses many issues and there are solutions for overcoming this problem. For performing operations on data stored inside the database system, users normally use SQL queries to retrieve those data. After retrieving perform various extractions and transformations on the dataset to make them suitable for application. Some approaches require demoralized table than normalized table. Because that contain more details than normalized tables and many analysis require analysis on large amount of data. There are four steps for the dataset preparation. Dataset preparation starts with data selection. In data selection, the analyst wants to perform analysis on the available data and select appropriate data for analysis. Second step is data integration. In data integration, data collected from different source are combined and stored inside a table. Third one is the data transformation. In data transformation the analyst wants to transform data into the format required for each operation. The last step is the data reduction. Here the data is compressed for the easiness of the analysis.Various approaches that used in dataset preparation are classified as shown in figure:

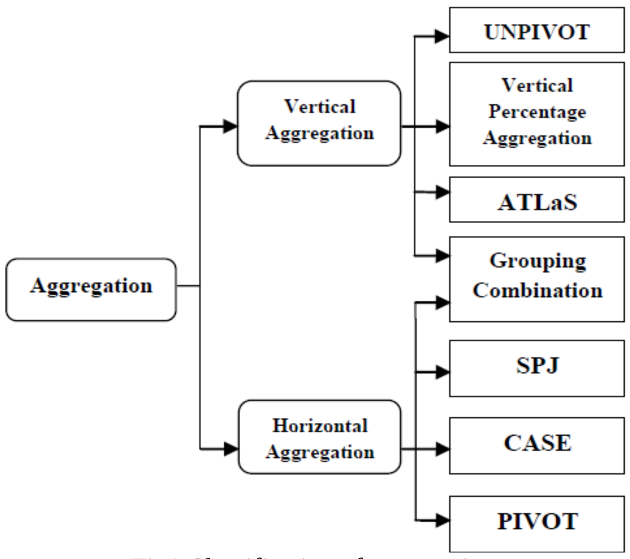


Fig1.Classification of aggregation

III. EXISTING SYSTEM

Creating a data set according to data mining requirements is time consuming and requires long SQL statements. Even though we are using automatically generated tool for data set, we have to customize dataset as per requirement. For this purpose we use joins and aggregations for creating data sets. We are focusing on aggregations. Most commonly used aggregations are sum of a column, average, maximum value, minimum value or row count and many aggregation functions and operations are available in SQL. All these aggregation functions have limitations like they returns scalar values. The main reason is, datasets stored in database come from On-Line Transaction Processing (OLTP) which is highly normalized. But data mining and machine learning algorithms requires elaborated aggregated form. Significant effort is needed to compute the aggregations on multi table structure. Standard aggregations are difficult to edit when they are performing on multiple tables which are having high cardinalities in many resultant row. To perform the analysis of databases on spreadsheets, it might be more comfort to having the aggregated functions on single group on single row. For example produce graphs, charts or compare datasets with repetitive information. OLAP tools generate SQL cod which transpose results more efficiently on aggregation and transposition mechanisms. Considering all the above conditions we introducing a new class of aggregate functions which are perform numeric and transpose results to generate horizontal data sets. Classes which are used to implement these functions are called horizontal aggregations. These horizontal aggregations produce added features of traditional SQL aggregations, that return a set of values in a horizontal layout, alternatively a scalar values

IV. PROPOSED SYSTEM

A new class of aggregations called horizontal aggregations are used to obtain result in horizontal layout. Since the existing SQL aggregation produce tables in vertical layout. The SQL code generation for horizontal aggregation is discussed follows.

B. SQL Code generation

To obtain horizontal aggregation, a small syntax extension to aggregate functions called in a select statement. This query will produce a table with y+1 column. The data are grouped based on the unique combination of values n1,n2,….,ny, and one aggregated value per group. To execute this query the query optimizer takes three input parameters.

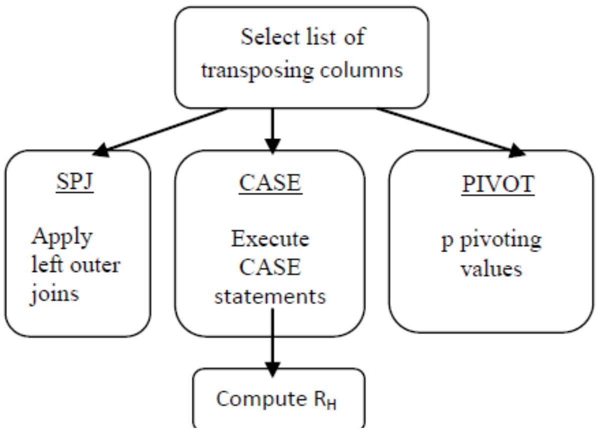
1) Input table, R
2) List of grouping attributes n1,n2,….,nt
3) Attribute to aggregate(X).

The aim of horizontal aggregation is to transpose aggregate attribute X by a column subset of n1,n2,….,nt. Assume that such subset is m1,m2,……,mv, where v<y. Or the GROUP BY list can be partitioned into two sub lists: one list contain n1,n2,….,nt and another list contain m1,m2,……,mv to transpose aggregated values. So, in a horizontal aggregation there are four input parameters to generate SQL code.

1) Input table F,
2) The list of GROUP BY attribute n1,n2,….,nt.
3) The attribute to aggregate(X).
4) The list of transposed columns m1,m2,……,mv

C. **Horizontal Evaluation Methods**

The existing three methods are used to evaluate horizontal aggregations. They are SPJ method, CASE method and PIVOT method. The SPJ method is based on relational operations in SQL. The CASE method uses the CASE construct in SQL. The PIVOT method uses the pivot built-in operator, which transforms rows to columns.



1) 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 Ri(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:

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

Table R0 identifies the number of rows in the final result and it became the primary key.

Tables R1,R2,….Rl contain the individual aggregations for each combination of m1,m2,…..,mV. { n1,n2,….,nt } become the key of RI.

SQL code for generating RI,

```
SELECT n1,n2,….,nt, Agg(X)
```

```
FROM R
WHERE m1=v1I AND……AND mvI
GROUP BY n1,n2,….,nt
```

The final SQL code for SPJ method,

```
SELECT R0.n1,R0.n2,……,R0.nt ,
R1.X, R2.X,……,Rl.X
FROM R0
LEFT OUTER JOIN R1
ON R0.n1 = R1.n1 and…and R0.nt = R1.nt
LEFT OUTER JOIN R2
ON R0.n1 = R2.n1 and… and R0.nt = R2.nt
…
LEFT OUTER JOIN Rl
ON R0.n1 = Rd.n1 and… and R0.nt = Rl.nt ;
```

2) CASE Method.

In this method, the CASE construct available in SQL is used. The case statement returns a value selected from a set of values based on Boolean expressions. 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. If there are no qualifying rows for the specific aggregate group then result must set to null as in the case of SPJ method. Agg () in the SQL code is the standard SQL aggregation that contain the case statement as the argument.

SQL Code for getting each distinct values of transposing columns,

```
SELECT DISTINCT m1,m2,……,mv
FROM R;
```

SQL Code for getting RH ,

```
SELECT n1,n2,….,nt
Agg (CASE WHEN m1 = v11 and …and mv = vv1
THEN X ELSE null END)…….
…..
Agg (CASE WHEN m1 = v1l and …and mv = vvl
THEN X ELSE null END)
FROM R
GROUP BY n1,n2,….,nt;
```

3) PIVOT Method

This method uses the built-in PIVOT operator in a Commercial DBMS, which transforms rows to columns. This operator can perform transposition, so it can be used for evaluating the horizontal aggregation. This method needs to determine how many columns are required to store the transposed table and it is combined with the GROUP BY clause. 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. If there are no qualifying rows for the specific aggregate group then result must set to null as in the case of SPJ method.

SQL code for getting the distinct values of transposing columns,

```
SELECT DISTINCT m1
FROM R;
```

SQL code for generating RH

```
SELECT n1,n2,….,nt ,
v1, v2,…,vl
INTO RH
FROM (SELECT n1,n2,….,nt,m1,X FROM R) Ru
PIVOT (
Agg (X) FOR m1 in (v1, v2,….,vl)
) AS P;
```

V. ADVANTAGES

The proposed horizontal aggregation has several advantages. By using the horizontal aggregation the manual work in preparation of data is reduced because there is no need to write the long SQL code by the data mining practitioner. Another advantage is, the SQL queries are written by person who has well knowledge about DBMS. So it is well efficient than the queries written by an end user. Next advantage is that the datasets are created in less time. By using the most efficient method that is, the CASE method reduces the time and space needed for processing the SQL query.

VI. CONCLUSION

Introduced a new class of extended aggregate functions, called horizontal aggregations which helps in preparing data sets for data mining and OLAP cube exploration. Specifically, horizontal aggregations are useful to create data sets with a horizontal layout, as commonly required by data mining algorithms and OLAP cross-tabulation. Basically, a horizontal aggregation returns a set of numbers instead of a single number for each group, resembling a multidimensional vector. Proposed an abstract, but minimal, extension to SQL standard aggregate functions to compute horizontal aggregations which just requires specifying sub grouping columns inside the aggregation function call. From a query optimization perspective, 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 SPJ method is important from a theoretical point of view because it is based on select, project, and join (SPJ) queries. The CASE method is most important contribution. It is in general the most efficient evaluation method and it has wide applicability since it can be programmed combining GROUP-BY and CASE statements. The three methods produce the same result. It is not possible to evaluate horizontal aggregations using standard SQL without either joins or “case” constructs using standard SQL operators.

REFERENCES

[1] 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.
[2] C. Ordonez and Z. Chen., Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining analysis, IEEE Transactions on Knowledge and Data Engineering (TKDE), 24(4), 2012.
[3] C. Ordonez, Data set preprocessing and transformation in a database system, Intelligent Data Analysis (IDA), 15(4), 2011.
[4] Srikanth Pasaragonda1 G. Charles Babu2,’ Prepare Datasets In SQL For Data Mining Analysis
[5] C. Ordonez, Data set preprocessing and transformation in a database system, Intelligent Data Analysis (IDA), 15(4), 2011.