Dataset Preparation and Indexing for Data Mining Analysis Using Horizontal Aggregations

Binomol George, Ambily Balaram

Abstract— **To analyze data efficiently, data mining systems are widely using datasets with columns in horizontal tabular layout. Preparing a dataset is more complex task in a data mining project, requires many SQL queries, joining tables and aggregating columns. In a relational database, a significant effort is required to prepare a summary dataset that can be used as input for the data mining process. Conventional RDBMS usually manage tables with vertical form. Tabular format takes suitable input to prepare datasets. But, existing SQL aggregations having limited capacity to prepare datasets because they return one column per aggregated group. In this paper, proposes simple, powerful methods to generate SQL code to return aggregated columns in a horizontal tabular layout. Aggregated columns in a horizontal tabular layout returns set of numbers, instead of one number per row. There are different methods to evaluate horizontal aggregations to represent an outline generate SQL code to return in a horizontal tabular format by using CASE, PIVOT and IIF methods. This class of new function is called as horizontal aggregation. Horizontal aggregations build datasets with a horizontal layout, which is the standard layout required by most data mining algorithms. After creating dataset in horizontal layout, create an index on the column fields, which will improve the queries firing to the prepared dataset.**

Index Terms— **aggregation, data mining, dataset generation, pivoting.**

I. Introduction

In a relational database environment with normalized tables, a significant effort is required to prepare a summary dataset [6] 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 [1] 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.

A. Advantages

There are several advantages for horizontal aggregation. First one is horizontal aggregation represent a template to generate SQL code from a data mining tool. This SQL code reduces manual work in the data preparation phase in data mining related project. Second is automatically generated code, which is more efficient than end user written SQL code. Thus datasets for the data mining projects can be created in less time. Third advantage is the datasets can be created entirely inside the DBMS. 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.

Manuscript received April 21, 2014.

***Binomol George**, Department of Computer Science & Engineering, Calicut University, Kozhikode, India, 9544393654*

***Ambily Balaram**, Department of Computer Science & Engineering, KMCT College of Engineering, Kozhikode, India, 9496218879.*

ISSN: 2278 – 7798

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.

III. HORIZONTAL AGGREGATION

A new class of aggregations that have similar behavior to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, 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 [3] called in a SELECT statement. Alternatively, horizontal aggregations can be used to generate SQL code from a data mining tool to build datasets for data mining analysis.

Our main goal is to define a template to generate SQL code combining aggregation and transposition (pivoting). A second goal is to extend the SELECT statement with a clause that combines transposition [5] with aggregation. In order to evaluate this query the query optimizer takes three input parameters: (1) the input table F, (2) the list of grouping columns L1,…..,Lm , (3) the column to aggregate (A). The basic goal of a horizontal aggregation is to transpose (pivot) the aggregated column A by a column subset of L1,…..,Lm; for simplicity assume such subset is R1,…….,Rk where k < m. In a horizontal aggregation there are four input parameters to generate
SQL code:

1) the input table F,
2) the list of GROUP BY columns L1,……..,Lj ,
3) the column to aggregate(A),
4) the list of transposing columns R1,…….,Rk.

A. Proposed Syntax in Extended SQL

Extend standard SQL aggregate functions with a transposing BY clause followed by a list of columns (i.e. R1; … ; Rk), to produce a horizontal set of numbers instead of one number. Our proposed syntax is as follows.

SELECT L1,…….,Lj, H(A BY R1,……,Rk)
FROM F
GROUP BY L1,……, Lj;

Here H() represents some SQL aggregation (e.g. sum(), count(), min(), max(), avg()). The function H() must have at least one argument represented by A, followed by a list of columns. The result rows are determined by columns L1,……,Lj in the GROUP BY clause if present. Result columns are determined by all potential combinations of columns R1,……,Rk, where k = 1 is the default. In order to get a consistent query evaluation it is necessary to use locking. The main reasons are that any insertion into F during evaluation may cause inconsistencies: (1) it can create extra columns in FH, for a new combination of R1,……, Rk; (2) it may change the number of rows of FH, for a new combination of L1,… ….,Lj ; (3) it may change actual aggregation values in FH.

The horizontal aggregation function H() returns not a single value, but a set of values for each group L1,……, LJ. Therefore, the result table FH must have as primary key the set of grouping columns {L1,…...,Lj} and as non-key columns all existing combinations of values R1,..…,Rk.

B. Examples

In a data mining project most of the effort is spent in preparing and cleaning a data set. A big part of this effort involves deriving metrics and coding categorical attributes from the data set in question and storing them in a tabular (observation, record) form for analysis so that they can be used by a data mining algorithm.

Assume, to summarize sales information with one store per row. In more detail, to know the number of transactions by store for each day of the week, the total sales for each

department of the store and total sales. The following query provides the answer.

```
SELECT
storeId,
sum(salesAmt BY dayofweekName),
count(distinct transactionid BY dayofweekNo),
sum(salesAmt BY deptIdName),
sum(salesAmt)
FROM transactionLine
,DimDayOfWeek,DimDepartment,DimMonth
WHERE transactionLine.dayOfWeekNo
=DimDayOfWeek.dayOfWeekNo
AND
transactionLine.deptId
=DimDepartment.deptId
GROUP BY storeId;
```

This query produces a result table like the one shown in Table1, Observe each horizontal aggregation effectively returns a set of columns as result and there is call to a standard vertical aggregation [2] with no subgrouping columns.

TABLE I
A MULTIDIMENSIONAL DATASET IN HORIZONTAL LAYOUT

	salesAmt				countTransactions				countItems				total
storeId	Mon	Tue	..	Sun	Jan	Feb	..	Dec	dairy	meat	produce	..	salesAmt
10	120	111		200	2011	1807		4200	34	57	101		25025
32	70	65		98	802	912		1632	32	65	204		14022
:													

For the first horizontal aggregation shows day names and for the second one shows the number of day of the week. These columns can be used for linear regression, clustering or factor analysis and also can analyze correlation of sales based on daily sales. Total sales can be predicted based on volume of items sold each day of the week. Stores can be clustered based on similar sales for each day of the week or similar sales in the same department.

IV. PROPOSED SYSTEM

The proposed framework comprises of importing databases, implementation of methods perform data transformation and generate horizontally aggregated datasets, saving the results, allows generation of mini-datasets and evaluation of methods. The proposed system architecture is as shown in figure 2.

The methods implemented are CASE, PIVOT and IIF. All the three methods generate datasets that are aggregated and are in horizontal form.

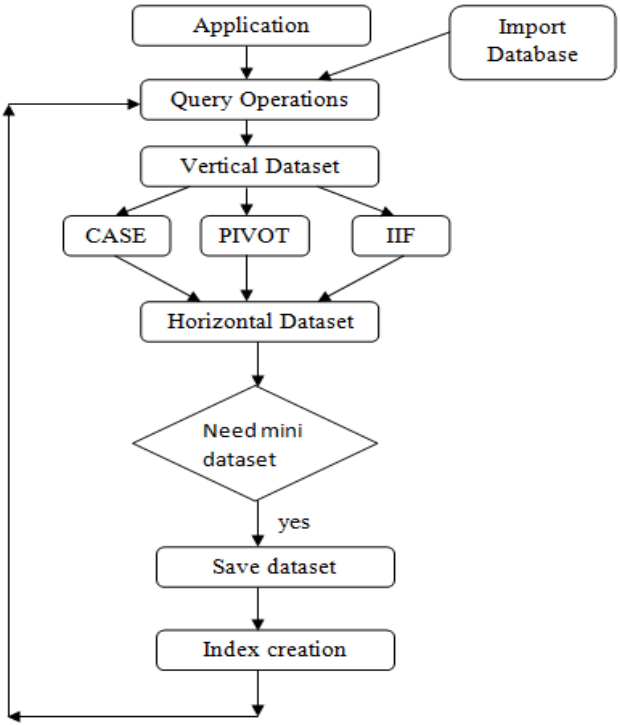

Fig 1: Proposed Methodology

As the proposed framework incorporates "Import database" facility, this makes system dynamic and also user can save in the output which is in horizontally aggregated form, to be saved in as a new relation in the same database.

A. *Query Evaluation Methods*

There are three basic strategies to evaluate horizontal aggregations. The first strategy relies on the SQL "case" construct; call it the CASE strategy. The second form uses the built-in PIVOT operator, which transforms rows to columns (e.g. transposing). The third method relies on the SQL "iif" statement, call it the IIF strategy.

CASE Method

CASE method can be performed by combining GROUP-BY and CASE statements. For this method, use the "case" programming construct available in SQL, where each non key value is given by a function that returns a number based on some conjunction of conditions. It is more efficient and has wide applicability. CASE statement evaluates the Boolean expression and return value from the selected set of values. CASE statement put the result to NULL when there is no matching row is found. This also produce resultant table in a horizontal layout. This will help us to minimize a lot of space used by user details. The general syntax for case as:

```
SELECT columns, Aggregate Function
(CASE WHEN Boolean expression THEN result ELSE
result expression END)
FROM table GROUP BY column.
```

There are two basic sub-strategies to compute FH. The first one directly aggregates from F and the second one computes

ISSN: 2278 – 7798

the vertical aggregation in a temporary table FV and then horizontal aggregations are indirectly computed from FV. The following statements compute FH:

```
SELECT DISTINCT R1,……,Rk
FROM FV;

INSERT INTO FH
SELECT L1,……,Lj
,sum(CASE WHEN R1 = v11 and.....and Rk = vk1
THEN A ELSE null END)
……
,sum(CASE WHEN R1 = v1d and…..and Rk = vkd
THEN A ELSE null END)
FROM FV
GROUP BY L1,L2,……,Lj;
```

PIVOT Method

The PIVOT operator which is a built-in operator in a commercial DBMS. Pivot is complementary data manipulation operators that modify the role of rows and columns in a relational table. In Pivot it transforms a series of rows into a series of fewer rows with additional columns. Pivoting refers to transposing rows data into columns.

```
SELECT DISTINCT R1
FROM F; /* produces v1,……..,vd */

SELECT
L1,L2,……..,Lj
,v1,v2,……..,vd
INTO FH
FROM (
SELECT L1,L2,……,Lj ,R1,A
FROM F) Ft
PIVOT(
V (A) FOR R1 in (v1,v2,……,vd)
) AS P;
```

IIF Method

IIF is a shorthand way for writing a CASE expression. It evaluates the Boolean expression passed as the first argument, and then returns either of the other two arguments based on the result of the evaluation. That is, the true_value is returned if the Boolean expression is true, and the false_value is returned if the Boolean expression is false or unknown. True_value and false_value can be of any type. The syntax for iif as:

```
IIF ( boolean_expression, true_value, false_value )
```

The following statements compute FH:

```
SELECT DISTINCT R1,……,Rk
FROM FV;

INSERT INTO FH
SELECT L1,……,Lj
,sum(IIF( R1 = v11,A,null))
……..
,sum(IIF(R1 = v1d,A,null))
FROM FV
GROUP BY L1,L2,……,Lj;
```

B. *Performance Evaluation*

CASE, PIVOT and IIF are methods that are used to generate horizontally aggregated datasets. In first method, CASE construct is used for transformation and generation of datasets. Using CASE, allows the user to classify the input data into category or groups. This helps in exploiting the data, in terms of class and titles. In PIVOT method, in-built SQL operator pivot is used. Using pivot operator allows transposing the input relation directly. In IIF method, returns one of two values, depending on whether the Boolean expression evaluates to true or false. All the three methods are proficient of generating same datasets as output provided same input along with same conditions are used. Performance of all the three methods is evaluated by observing time taken by each method to generate horizontally aggregated dataset for various inputs. On the whole, the evaluation leads to specifics listed: as the number of rows and columns increase, time taken by methods gradually increases. Most imperative is that as input change (due to import database feature) the SQL queries also vary for each of the given methods which impinge on the recital of all three input methods.

V. CONCLUSION

We introduced a new class of extended aggregate functions, called horizontal aggregations which help preparing datasets for data mining analysis. Specifically, horizontal aggregations are useful to create datasets with a horizontal layout, as commonly required by data mining algorithms and OLAP cross-tabulation. Data transformation is an imperative stage of knowledge discovery process. Data transformation yields output which is considered as a better input for data mining process. Standard SQL provides aggregation function which generates output as a single row and no increase in number of columns. The three methods demonstrated in proposed framework CASE, PIVOT and IIF, overcome these shortcomings of SQL vertical aggregation. CASE and IIF methods are traditionally not in-built function/feature of SQL to generate horizontal aggregated columns; whereas PIVOT uses an in-built feature of SQL in grouping along aggregation function. CASE, PIVOT and IIF methods for data transformation provide the provision to generate the datasets with horizontal aggregation i.e. new columns which originally did not exist in input. Aggregation is performed for these newly generated columns along with the existing columns. The implementation of CASE, PIVOT and IIF in this framework reduces complexity of writing SQL query for the user to generate horizontal dataset. Performance of all the three methods is evaluated by observing time taken by each method to generate horizontally aggregated dataset for various inputs. Fundamentally IIF generates output in

least possible time and can thought of as a fastest method out
of three. 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.

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. *Vertical and horizontal percentage aggregations*, In Proc. ACM SIGMOD Conference, pages 866–871, 2004.

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

[4] 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.

[5] C. Ordonez. *Horizontal aggregations for building tabular data sets*, In Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop, pages 35.42, 2004.

[6] C. Ordonez, *Data set preprocessing and transformation in a database system*, Intelligent Data Analysis (IDA), 15(4), 2011.

[7] A. Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, A. Gupta, L. Sheng, and S. Subramanian. *Spreadsheets in RDBMS for OLAP*. In Proc. ACM SIGMOD Conference, pages 52–63, 2003.

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

First Author

Binomol George received her B.E. degree from SCAD College of Engineering and Technology, Anna University, and Tamilnadu state, India in the year 2012. Currently, doing her M.Tech. in Computer Engineering from KMCT College of engineering, Calicut, Kerala state, India. Her research interests include in Data mining, Networks.

ISSN: 2278 – 7798