Volume 7, Issue 5, May 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey on Preparing Data Sets for Data Mining Analysis using Horizontal Aggregations in SQL

Prashant B. Rajole
Department of Computer Engineering, MCOERC, Nasik,
Maharashtra, India

DOI: 10.23956/ijarcsse/V7I4/0199

Abstract— Data mining is the field which has effectiveness in real world scenarios. Data sets are prepared from accepted transactional databases for the purpose of data mining. A vast amount of time is needed for creating the dataset for the data mining analysis because data mining developers required to write multifaceted SQL queries and many tables are to be coupled to get the aggregated result. Here, we recommended simple, however powerful, techniques to generate SQL code to formulate aggregated columns in a very horizontal tabular page layout, getting a few numbers as opposed to one variety per short period. This new functions class is named horizontal aggregations. Data sets are build using horizontal aggregations with a horizontal de-normalized layout (e.g., observation- variable, point dimension, instance-feature) which is the standard layout required by most data mining algorithms. Building user-defined new aggregate function that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout is focused in this paper. Horizontal aggregations represent an ex-tended form of traditional SQL aggregations in which it returns a set of values in a horizontal layout. It is a new class of aggregations that have similar behavior to SQL standard aggregations which produces tables with a horizontal layout. As compare to standard SQL aggregations we call it vertical aggregations since they produce tables with a vertical layout. In Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement. In other words, horizontal aggregations can be used to generate SQL code from a data mining tool to build data sets for data mining analysis.

Keywords—Data Mining, Aggregation, SQL, Pivot, Case

I. INTRODUCTION

Data mining is mostly an increasingly important technology for extracting useful knowledge hidden in large collections of data. The task of preparing a data set for analysis is generally the most time consuming task in a data mining project which requires many complex SQL queries, aggregating columns and joining tables. Existing system have many limitation to prepare datasets because it return one column per aggregated group. To solve this difficulty here we recommend, simple and powerful method to generate SQL code which return aggregated columns in horizontal layout which return the set of numbers instead of one number per row.

So building a suitable data set for data mining purposes is a time-consuming task. It requires writing long SQL statements or customizing SQL code if it is generated by some tool automatically. There is two main ingredients in such SQL code: joins and aggregations here we focus on the second method. Unfortunately, all these aggregations have limitations to build data sets for data mining purposes because 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 normalized highly. But data mining, statistical, or machine learning algorithms generally needed aggregated data in summarized form.

Standard aggregations are hard to interpret when there are many result rows, mostly when cardinalities of grouping attributes are high. To perform analysis of exported tables into spreadsheets it may be more convenient to have aggregations on the same group in one row. Because of such limitation, this paper proposes a new aggregate function that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout. Horizontal aggregations represent an ex-tended form of traditional SQL aggregations in which return a set of values in a horizontal layout.

Challenges involved in developed Systems:

1. Preparing a data set for analysis is time consuming and complex.
2. Existing system return one column per aggregated group.

The rest of the paper is structured as follows: Section 2 summarizes the existing algorithms to solve problems of preparing data sets for data mining analysis. Section 3 describes the implementation details to present the knowledge and various algorithms used in our work with mathematical model; it also describes the design of our projected system. The

results obtained by our work are given, in detail, in Section 4. Section 5 summarizes conclusion and future scope of the projected system.

Here we presented three methods to evaluate horizontal aggregations.

SPJ –This method relies only on relational operations i.e. only doing select, join, project, and aggregation queries.

CASE -The second form relies on the SQL "case" constructs. Here each table has an index on its primary key for efficient join processing.

PIVOT – This method uses the built-in PIVOT operator, which transforms rows to columns (e.g., transposing).

Main objectives of the system:

- To design an algorithm for the data set analysis in data mining.
- To provide a general framework for horizontal aggregations like SPJ, PIVOT, and CASE.
- To implement method to analyze data mining for preparing data sets using horizontal aggregations.

The proposed work aims at Preparing Data Sets for Data Mining Analysis using Horizontal Aggregations in SQL.

II. LITERATURE SURVEY

There exist many proposals that have extended SQL syntax. The closest data mining problem related to OLAP processing is association rule mining [1]. SQL extensions used to define aggregate functions for association rule mining are introduced in [2]. In this case, the goal is to efficiently & effectively compute item set support. Regrettably , there is no notion of transposing results since transactions are given in a vertical layout. A clustering algorithm programming with SQL queries is explored in [3], which shows a horizontal layout of the data set enables easier & simple SQL queries. To perform spreadsheet-like operations alternative SQL extensions were introduced in [4]. Their optimizations have the purpose of avoiding joins to express cell formulas but they are not optimized to perform partial transposition for each group of result rows. The PIVOT and CASE methods also avoid joins as well. Several SQL primitive operators for transforming data sets for data mining were introduced in [5]; the more similar one to ours is an operator to transpose a table, based on one chosen column. The TRANSPOSE operator [5] is identical to the unpivot operator, producing several rows for one input row. The major or significant difference is that, as 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 is inverse operators with respect to horizontal aggregations. A vertical layout may give more flexibility expressing data mining computations (e.g., decision trees) with SQL aggregations and group-by queries, but it is less efficient as compared to horizontal layout.

Horizontal aggregations are closely relate to horizontal percentage aggregations [6][7]. The dissimilarity between both approaches are that percentage aggregations require aggregating at two grouping levels, require dividing numbers and need taking care of numerical issues (For e.g. dividing by zero). Horizontal aggregations are more general, have wider applicability and in fact, they can be used as a primitive extended operator to compute percentages.

Outcomes/Conclusion of literature survey:

Studying the existing data mining analysis algorithms for preparing data sets, we conclude with the aim of

- Finding an efficient algorithm for the data set analysis in data mining.
- Providing a general framework for horizontal aggregations like SPJ, PIVOT, CASE.
- Preparing data sets for data mining analyzing.

III. IMPLEMENTATION DETAILS

This section presents a software and hardware requirements for the project implementation. Platform used for project implementation is given below:

3.1 Algorithm:

In existing system data mining algorithms that can directly analyze data sets having a vertical layout (e.g., in transaction format), 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). Existing system is more time consuming process to prepare a dataset [8]. It requires many complex SQL queries, Joining tables and aggregation columns. Today's existing SQL aggregations have restriction i.e. limitations to prepare data sets because they return one column per aggregated group [9].

In this paper, propose a new aggregate function that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout. Horizontal aggregations more likely represent an ex-tended form of traditional SQL aggregations, which return a set of values in a horizontal layout [10][11]. It is a new class of aggregations that have similar behavior to SQL standard aggregations, but which produce tables with a horizontal layout. As compare to standard SQL aggregations we call it vertical aggregations since they produce tables with a vertical layout [12]. Horizontal aggregations just necessarily require a small syntax extension to aggregate functions called in a SELECT statement. Alternatively, horizontal aggregations can be used to produce SQL code from a data mining tool to build data sets for data mining analysis.

Template generation:

This paper proposed horizontal aggregations to provide several unique features and advantages. At first, they represent a template to generate SQL code from a data mining tool. Such SQL code automates writing SQL queries,

optimizing them, & also testing them for correctness. This SQL code mostly reduces manual work in the data preparation phase in a data mining project.

Query generation:

Since SQL code is generated automatically it is likely to be more effective as well as efficient than SQL code written by an end user. Therefore, data sets can be created in less time.

Query Syntax Analysis:

The SQL code is automatically generated so it must need to check the syntax before using that piece of code.

Data Extraction:

The data set can be created completely inside the DBMS. In modern database environments, it is common to export de-normalized data sets to be further cleaned and transformed outside a DBMS in external tools (For e.g. statistical packages). In modern database environments, it is common to export denormalized data sets to be cleaned and transformed outside a DBMS in external tools, exporting large tables outside a DBMS is slow, creates inconsistent copies of the same data and compromises database security. Therefore, in this paper, provide a more efficient, better integrated and more secure solution compared to external data mining tools. Horizontal aggregations mostly require a small syntax extension to aggregate functions called in a SELECT statement. Horizontal aggregations which can be used to generate SQL code from a data mining tool to build data sets for data mining analysis.

3.2 Design detail of the proposed System:

Following Figure shows a detailed flow of algorithm execution which comprises

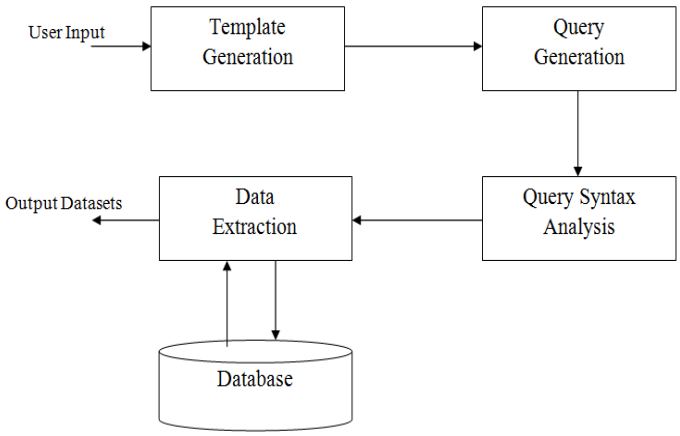

Figure 1: Architectural Diagram

This proposed system provides several unique features and advantages. Firstly, it represents a template to generate SQL code from a data mining tool. Such SQL code automates writing SQL queries, optimizing them, and also testing them for correctness & accuracy which reduces manual work in the data preparation phase in a data mining project. Second, since SQL code is generated automatically it is likely to be more efficient than SQL code written by an end user. Suppose a person who does not know SQL well or someone who is not familiar with the database schema. Therefore, data sets can be generated in less time. The data set can be created completely inside the DBMS.

3.3 Mathematical Modeling:

System for analyzing data sets using horizontal aggregation technique illustrates the mathematical model that contains columns. The transactional Dataset which consist of following entities:

Input: Database

Output: Data sets

Function F1→ Consider input dataset.

Let, Database consist of columns such as C1, C2,..Cn.

Database is given as an input to the Horizontal aggregation methods viz; CASE, PSJ, PIVOT.

Function F2 → Template Generation using SPJ method.

1. SPJ Method: - This method based on standard relational algebra operators (SPJ queries).
   - Creates one table with a vertical aggregation for each result column.
   - Join all those tables to generate another table.

   SPJ {c1, c2,..cn }→ S1, S2,..Sn.

   Where as S is output of SPJ method.

   → S € SPJ.

Function F3 → Template Generation using CASE method.

2. CASE Method: - This method used in any statement or clause that allows a valid expression.
   - Returns a value which is selected from a set of values based on Boolean expression.
   - Query evaluation needs to be combine the desired aggregation with "case" statement for each & every distinct combination of values of R1; R2……..,Rk.

   CASE {C1, C2 ,..Cn}→ R1, R2 ,…Rn. where R is output of CASE method.

➔ R € CASE.

Function F4 → Template Generation using PIVOT method.

3. PIVOT Method: -In this method, a built-in operator which transforms row to columns.

- needs to determine how many columns are require to store the transposed table and also it can be combined with the GROUP BY clause. The optimized PIVOT method SQL is given as follows:

PIVOT {C1, C2,..Cn}→L1,L2,..Ln. where L is output of PIVOT method.

→ L € PIVOT.

Function F5 → Query Generation and Query syntax analysis.

Query Set (QS) is generated by considering all outputs of above three methods gives result sets (DS) as:

QS{S, R, L} →DS1, DS2,..DSn. € DS

Function F6 → Data Extraction.

Output/outcome: - Data sets € DS.

Table: Functional Dependency table

	F1	F2	F3	F4	F5	F6
F1	1	0	0	0	0	0
F2	1	1	0	0	0	0
F3	0	1	1	0	0	0
F4	0	0	1	1	0	0
F5	1	1	1	1	1	0
F6	0	0	0	0	1	1

IV. RESULTS

This paper creates product application for the user. Firstly, create admin login form and user login form. Admin module can perform entries in database like branch registration, employee registration, and product registration and sell entry. Admin only can update all registered data. Employee can view product, sell and also update product, sell. User on the other hand can only view the products information and sell information in different views like SPJ view, CASE view, and Pivot view.

Sell details can be aggregated by SPJ method, CASE method and PIVOT method. PIVOT method can be transposing the rows into aggregated columns. This paper helps for login the user account, view products and sell details. All this information aggregated in horizontal tabular format by SPJ, CASE and PIVOT wise method. We believe this is remarkable paper, since our presentation is based on generating SQL code and not modifying the query optimizer internally. Although, both CASE and PIVOT evaluation methods are more significantly faster than the SPJ method.

V. CONCLUSION AND FUTURE SCOPE

In this paper, discussed about SQL code which automatically generates it is likely to be more efficient than SQL code written by an end user. Therefore, data sets can be produce in less time. Horizontal aggregations just needed a small syntax extension to aggregate functions called in a SELECT statement.

In future, this work can be extended to develop a more formal model of evaluation methods to achieve better results. Further we can also develop more complete I/O cost models.

REFERENCES

[1] S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '98), pp. 343-354, 1998.

[2] H. Wang, C. Zaniolo, and C.R. Luo, "ATLAS: A Small But Complete SQL Extension for Data Mining and Data Streams," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03), pp. 1113- 1116, 2003.

[3] Ordonez, "Integrating K-Means Clustering with a Relational DBMS Using SQL," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 2, pp. 188-201, Feb. 2006.

[4] Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N.Folkert, A. Gupta, L. Sheng, and S. Subramanian, "Spreadsheets in RDBMS for OLAP," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 52-63, 2003.

[5] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman, "Non- Stop SQL/MX Primitives for Knowledge Discovery," Proc. ACM SIGKDD Fifth Int'l Conf. Knowledge Discovery and Data Mining (KDD '99), pp. 425-429, 1999.

[6] Ordonez,"Vertical and Horizontal Percentage Aggregations," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '04), pp. 866-871, 2004.

[7] C. Ordonez, "Statistical Model Computation with UDFs," IEEE Trans. Knowledge and Data Eng., vol. 22, no. 12, pp.1752-1765, Dec. 2010

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

[9] E.F. Codd, "Extending the Database Relational Model to Capture More Meaning," ACM Trans. Database Systems, vol. 4, no. 4, pp. 397-434, 1979.

[10] Rajesh Reddy Muley, Sravani Achanta and Prof.S.V.Achutha Rao, "Query Optimization Approach in SQL to prepare Data Sets for Data Mining Analysis", International Journal of Computer Trends and Technology (IJCTT) – vol.4, no.8, pp 1-5,August 2013.

[11] J.A. Blakeley, V. Rao, I. Kunen, A. Prout, M. Henaire, and C. Kleinerman, ".NET Database Programmability and Extensibility in Microsoft SQL Server," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), pp. 1087-1098, 2008.

[12] C. Ordonez, "Vertical and Horizontal Percentage Aggregations," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'04), pp. 866-871, 2004.

