Volume 2, Issue 11, November 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Experimental Analysis using PIVOT Method in Data Mining

A.Lakshman Rao[1]
Associate Professor
Department of CSE
Pragati Engineering College
Surampalem, E.G. District

V.V.Satyanarayana Murty.S[2]
Final M.Tech Student
Department of CSE
Pragati Engineering College,
Surampalem, E.G. District

Abstract: ***Data mining is widely used domain for extracting trends or patterns from historical data .To analyze 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. Relational databases are acceptable repository for structured data; integrating data mining algorithms with a relational DBMS is an essential research issue for database programmers. In a relational database, a significant effort is required to prepare a summary data set that can be used as input for the data mining process.***

Key Words: Data mining, SQL, data set generation, OLAP.

I. 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. That is the case with models like clustering, classification; regression and PCA consult [3]. 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 article introduces a new class of aggregate functions that can be used to build data sets in a horizontal layout (denormalized with aggregations), automating SQL query writing and extending SQL capabilities. We show evaluating horizontal aggregations is a challenging and interesting problem and we introduced alternative methods and optimizations for their efficient evaluation[6]. 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[2].

Some other aggregations return the average, maximum, minimum or row count over groups of rows. There exist many aggregations 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[8]. 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[1]. 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[4]. 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. Literature survey

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both[1]. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support[3]. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration r taken into account for developing the proposed system. As horizontal aggregations are capable of producing data sets that can be used for real world data mining activities.

This paper presents three horizontal aggregations namely SPJ, PIVOT and CASE. It does mean that we enhanced these operators that are provided by SQL in one way or the other. The SPJ aggregation is developed using construct with standard SQL operations. PIVOT makes use of built in pivoting facility in SQL while the CASE is built based on the SQL CASE construct. We have built a web based prototype application that demonstrates the effectiveness of these constructs[9]. The empirical results revealed that these operations are able to produce data sets with horizontal layout that is suitable for OLAP operations or data mining operations. The motivation behind this work is that developing data sets for data mining operations is very difficult and time consuming. One problem in this area is that the existing SQL aggregations provide a single row output. This output is not suitable for data mining operations. For this reason we have extended the functionalities of CASE, SPJ and PIVOT operators in such a way that they produce data sets with horizontal layouts[10].

III. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective[11]. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

IV. Implementation of Admin Module

Admin will upload new connection form based on regulations in various states. Admin will be able upload various details regarding user bills like a new connection to a new user, amount paid or payable by user. In case of payment various details regarding payment will be entered and separate username and password will be provided to users in large.

Implementation of User Module

User will be able to view his bill details on any date may be after a month or after months or years and also he can to view the our bill details in a various ways for instance, The year wise bills, Month wise bills, totally paid to bill in EB. This will reduce the cost of transaction. If user thinks that his password is insecure, he has option to change it. He also can view the registration details and allowed to change or edit and save it.

V. Implementation of View Module

Admin has three ways to view the user bill details, the 3 ways are

i) SPJ
ii) PIVOT
iii) CASE

SPJ : While using SPJ the viewing and processing time of user bills is reduced.

PIVOT : This is used to draw the user details in a customized table. This table will elaborate us on the various bill details regarding the user on monthly basis.

CASE: using CASE query we can customize the present table and column based on the conditions. This will help us to reduce enormous amount of space used by various user bill details. It can be viewed in two difference ways namely

Horizontal and Vertical. In case of vertical the number of rows will be reduced to such an extent it is needed and column will remain the same on other hand the Horizontal will reduce rows as same as vertical and will also increase the columnar format.

VI. Proposed Approach Advantages

- The SQL code reduces manual work in the data preparation phase in a data mining project.
- The SQL code is automatically generated it is likely to be more efficient than SQL code written by an end user.
- The data sets can be created in less time.
- The data set can be created entirely inside the DBMS.

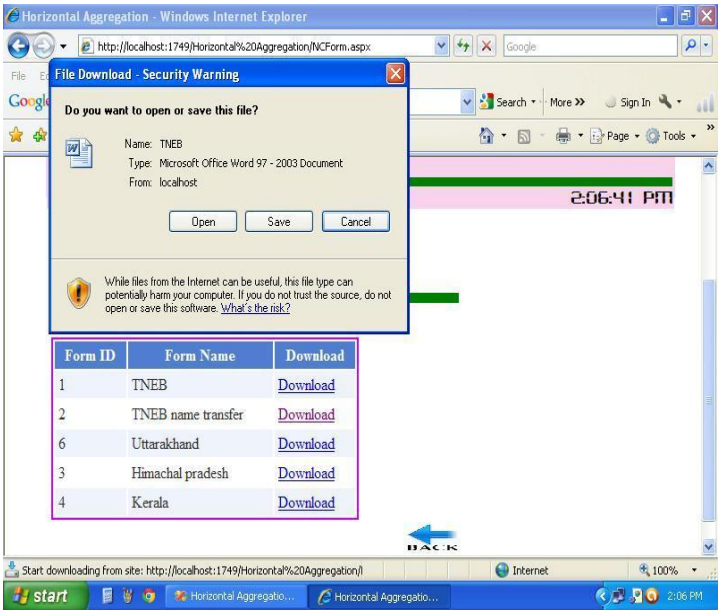


Figure 1: Opening the File

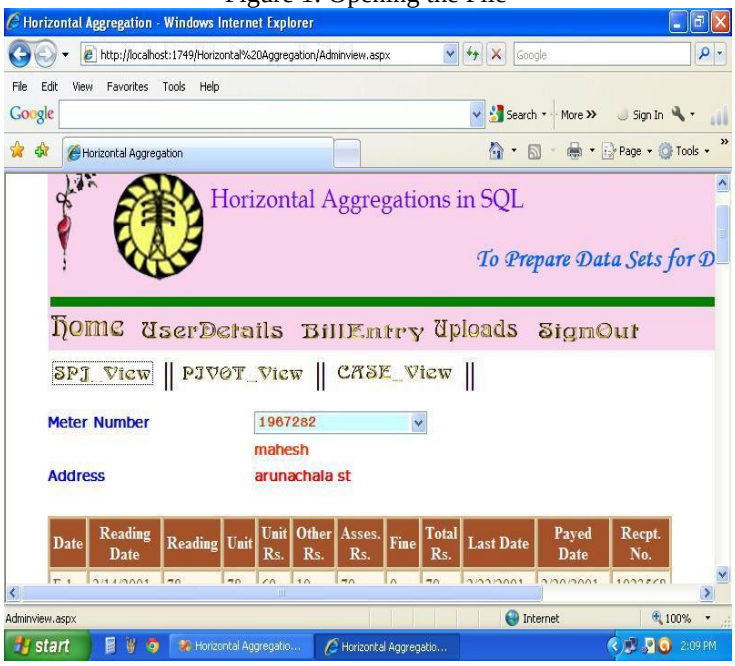
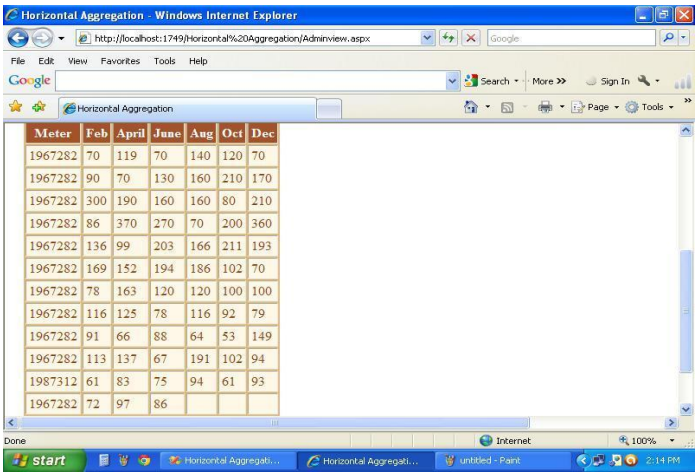


Figure 2: User data details



Meter	Feb	April	June	Aug	Oct	Dec
1967282	70	119	70	140	120	70
1967282	90	70	130	160	210	170
1967282	300	190	160	160	80	210
1967282	86	370	270	70	200	360
1967282	136	99	203	166	211	193
1967282	169	152	194	186	102	70
1967282	78	163	120	120	100	100
1967282	116	125	78	116	92	79
1967282	91	66	88	64	53	149
1967282	113	137	67	191	102	94
1987312	61	83	75	94	61	93
1967282	72	97	86			

Figure 3: User records to the dataset

Figure 4: Bill payment slip

VII. Conclusion:

In this paper we extended three aggregate functions such as CASE, SPJ and PIVOT. These are known as horizontal aggregations. We have achieved it by writing underlying constructs for each operator. When they are used, internally the corresponding construct gets executed and the resultant data set is meant for OLAP (Online Analytical Processing). Vertical aggregations such as SUM, MIN, MAX, COUNT, and AVG return a single value output. However, that output can't be used for data mining operations. In order to prepare real world datasets that are very much suitable for data mining operations, we explored horizontal aggregations by developing constructs in the form of operators such as CASE, SPJ and PIVOT. We have explained it is not possible to evaluate horizontal aggregations using standard SQL without either joins or "case" constructs using standard SQL operators. Our proposed horizontal aggregations can be used as a database method to automatically generate efficient SQL queries with three sets of parameters: grouping columns, sub grouping columns and aggregated column.

The fact that the output horizontal columns are not available when the query is parsed (when the query plan is explored and chosen) makes its evaluation through standard SQL mechanisms infeasible. Our experiments with large tables show our proposed horizontal aggregations evaluated with the CASE method have similar performance to the built-in PIVOT operator. We believe this is remarkable since our proposal is based on generating SQL code and not on internally modifying the query optimizer. Both CASE and PIVOT evaluation methods are significantly faster than the SPJ method. Precomputing a cube on selected dimensions produced acceleration on all methods.

References

[1] C. Ordonez, "Data Set Preprocessing and Transformation in a Database System," Intelligent Data Analysis, vol. 15, no. 4, pp. 613- 631, 2011.

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

[3] C. Ordonez and S. Pitchaimalai, "Bayesian Classifiers Programmed in SQL," IEEE Trans. Knowledge and Data Eng., vol. 22, no. 1 , pp. 139-144, Jan. 2010.

[4] J. Han and M. Kamber, Data Mining: Concepts and Techniques, first ed. Morgan Kaufmann, 2001.

[5] C. 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.

[6] 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.

[7] 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.

[8] A. 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.

[9] H. Garcia-Molina, J.D. Ullman, and J. Widom, Database Systems: The Complete Book, first ed. Prentice Hall, 2001.

[10] C. Galindo-Legaria and A. Rosenthal, "Outer Join Simplification and Reordering for Query Optimization," ACM Trans. Database Systems, vol. 22, no. 1, pp. 43-73, 1997.

[11] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: alternatives and implications. In *Proc. ACM SIGMOD Conference*, pages 343–354, 1998.

[12] 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.

