Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis

Gangamma.G.Hediyalad,
M.Tech(Computer Science and Engineering),
Bapuji Institute of Engineering and Technology,
Davangere,India
gangahediyalad@gmail.com

Vinutha HP
Assistant Professor,CS&E Dept.,
Bapuji Institute of Engineering and Technology,
Davangere,India
vinuprasad.hp@gmail.com

***Abstract*— Data mining is widely used domain for extracting trends or patterns from historical data. However, the databases used by enterprises can't be directly used for data mining. It does mean that Data sets are to be prepared from real world database to make them suitable for particular data mining operations. However, preparing datasets for analyzing data is tedious task as it involves many aggregating columns, complex joins, and SQL queries with sub queries. More over the existing aggregations performed through SQL functions such as MIN, MAX, COUNT, SUM, AVG return a single value output which is not suitable for making datasets meant for data mining. In fact these aggregate functions are generating vertical aggregations. This paper presents techniques to support horizontal aggregations through SQL queries. The result of the queries is the data which is suitable for data mining operations. It does mean that this paper achieves horizontal aggregations through some constructs built that includes SQL queries as well. The methods prepared by this paper include CASE, SPJ and PIVOT. We have developed a prototype application and the empirical results reveal that these constructs are capable of generating data sets that can be used for further data mining operations.**

Keywords-Aggregations, SQL, data mining, OLAP, and data set generations

I. INTRODUCTION

RDBMS has become a de facto standard for storing and retrieving large amount of data. This data is permanently stored and retrieved through front end applications. The applications can use SQL to interact with relational databases. Preparing databases needs identification of relevant data and then normalizing the tables.Aggregations are supported by SQL to obtain summary of data. The aggregate functions supported by SQL are SUM, MIN, MAX, COUNT and AVG. These functions produce single value output. These are known as vertical aggregations. This is because each function operates on the values of a domain vertically and produces a single value result. The result of vertical aggregations is useful in calculations or computations. However, they can't be directly used in data mining operations further. In fact summary data sets can be prepared and they can be used further in data mining operations [1]. The summary data can also be used in statistical algorithms [2],[3]. Most of the data mining operations expect a data set with horizontal layout with many tuples and one variable or dimension per column. This is the case with many data mining algorithms like PCA, regression, classification, and clustering [4], [2].

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. 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.

The proposed horizontal aggregations have some unique features and they are very useful. The advantages include, they provide construct that can generate SQL code that results in dataset which is suitable for data mining; the SQL code supports automation of writing SQL queries, testing them and optimizing them. As the proposed constructs are based on SQL, it reduces lot of coding as it is a powerful data retrieval language. The proposed system is user friendly as users are never expected to write queries. Instead, they just make queries in a user-friendly fashion. End users who do not know SQL also can make use of the proposed system. As SQL code is automatically generated based on the operator used in the query, it reduces lot of manual mistakes. In modern database where data is stored because of day to day operations, users do not get a chance to use data directly for mining operations. Instead, it has to be transformed to make sense for data mining operations. Generally data from business database is converted, and loaded into some other data sets known as data warehouse. The proposed horizontal aggregations can be used to generate data sets for the purpose of data mining analysis.

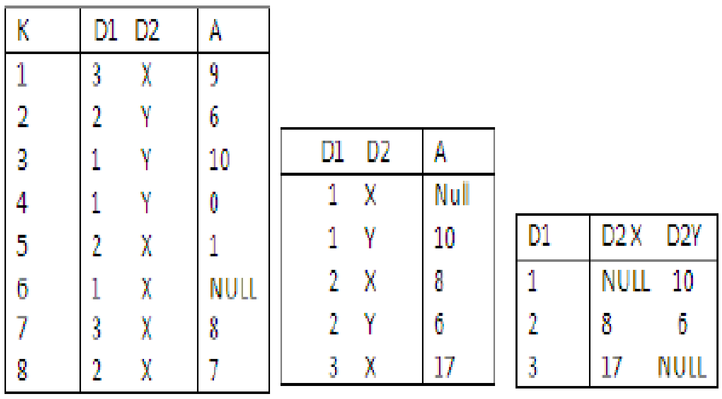
II. RELATED WORK

SQL has been around since its inception and being used widely for interacting with relational databases both for storing and retrieving data. The SQL provides all kinds of constructs such as projections, selections, aggregations, joins and sub queries. Query optimization and using the result of query further is an essential task in database operations. As part of queries, aggregations are used to get summary of data. Aggregate functions such as SUM, MIN, MAX, COUNT, and AVG are used for obtaining summary of data [5]. These aggregations produce a single value output and can't provide data in horizontal layout which can be used for data mining operations. In other words, the vertical aggregations can't produce data sets for data mining. Association rule mining is one of the problems pertaining to OLAP processing [6]. SQL aggregate functions are extended for the purpose of association rule mining in [7]. The aim of this is to support data mining operations efficiently. The drawback of this is that it is not capable of producing results in tabular format with horizontal layout convenient for data mining operations. In [5] a clustering algorithm is explored which makes use of SQL queries internally. It is capable of showing horizontal layout for further mining operations. For performing spreadsheet like operations, alternative SQL extensions are proposed in [8]. They have optimizations too for joins and they do not have optimizations for partial transposition of resultant groups. Joins can be avoided using CASE and PIVOT constructs. Traditional relational algebra [9] has to be adapted to generate new class of aggregations known as horizontal aggregations for generating data sets for data mining operations. This is the focus of our work. The problem of optimizing outer joins is presented in [10]. However, it is not suitable for large queries. Traditional query optimizations [11] use tree-based plans for optimization. This is similar to SPJ method. CASE is also used with SQL optimizations. PIVOT in sql is used for pivoting results. Lot of research has been around on aggregations and optimizations of SQL operations. They also include cross tabulation and explored much in [12] in case of cube queries. Unpivoting relational tables is also explored in [13] where each input row is used to calculate the decision trees. The result contains multiple rows with attribute – value pairs that behave like an inverse operator for horizontal aggregations. Many SQL operators are available to transform data from one format to another format [14]. The TRANSPOSE operator is similar to unpivot operator which produces many rows for each input row. TRANSPOSE can reduce the number of operations when compared with PIVOT. These two are having inverse relationship as the results are proving this. For data mining operations that produce decision trees, vertical aggregations can be used while the horizontal aggregations produce more convenient horizontal layout that is best suited for data mining operations. In SQL Server [15] both pivot and unpivot operations are made available.

Horizontal aggregations are explored to some extent in [16] and [17] with some limitations. It does mean that the result of these can't be efficiently used for further data mining operations. The proposed horizontal aggregations are different from the built in aggregations that come with SQL. Our operators such as CASE, PIVOT and SPJ are extensions to corresponding SQL operators. For instance CASE is our programming construct that is based on the CASE of SQL; PIVOT is our programming construct that is based on SQL pivoting operation; and the SPJ construct is built using standard SQL queries only.

III. HORIZONTAL AGGREGATIONS

For describing the methods pertaining to the proposed horizontal aggregations such as PIVOT, CASE and SPJ, input table as shown in Fig. 1 (a), traditional vertical sum aggregation as shown in Fig 1 (b) and horizontal aggregation as shown in Fig. 1 (c) are considered.



K	D1	D2	A
1	3	X	9
2	2	Y	6
3	1	Y	10
4	1	Y	0
5	2	X	1
6	1	X	NULL
7	3	X	8
8	2	X	7

D1	D2	A
1	X	Null
1	Y	10
2	X	8
2	Y	6
3	X	17

D1	D2X	D2Y
1	NULL	10
2	8	6
3	17	NULL

Fig. 1 – Input table (a), traditional vertical aggregation (b), and horizontal aggregation (c)

As can be seen in fig.1, input table has some sample data. Traditional vertical sum aggregations are presented in (b) which is the result of SQL SUM function while (c) holds the horizontal aggregation which is the result of SUM function.

IV. STEPS USED IN ALL METHODS

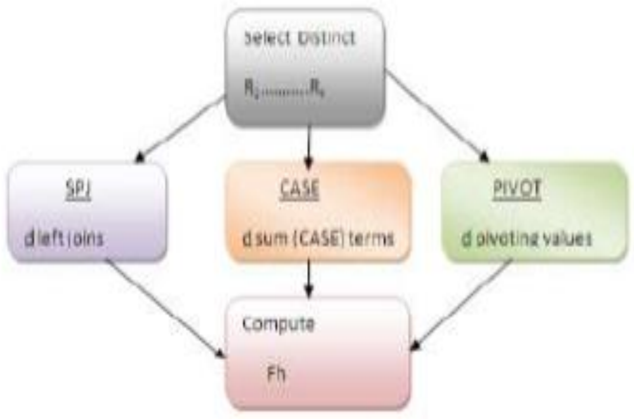


Fig. 2 shows steps on all methods based on input table

As can be seen in fig.2, for all methods such as SPJ, CASE and PIVOT steps are given. For every method the procedure starts with SELECT query. Afterwards, corresponding operator through underlying construct is applied. Then horizontal aggregation is computed.

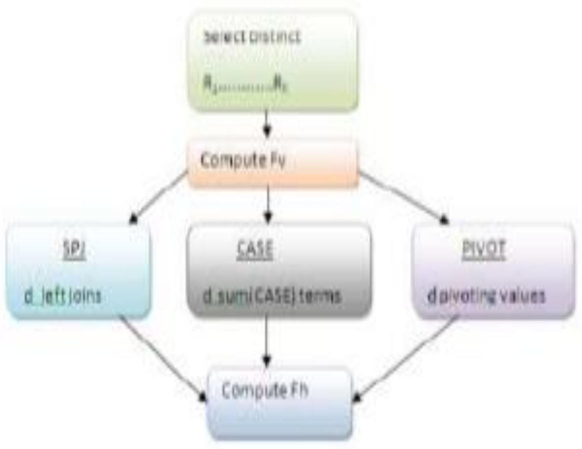


Fig.3 shows steps on all methods based on table containing results of vertical aggregations

As can be seen in fig.3, for all methods such as SPJ, CASE and PIVOT steps are given. For every method the procedure starts with SELECT query. Afterwards, corresponding operator through underlying construct is applied. Then horizontal aggregation is computed.

1 SPJ Method

This method is based on the relational operators only. In this method one table is created with vertical aggregation for each column. Then all such tables are joined in order to generate a tbale containing horizontal aggregations. The actual implementation is based on the details given in [18].

2 PIVOT Method

This aggregation is based on the PIVOT operator available in RDBMS. As it can provide transpositions, it can be used to evaluate horizontal aggregations. PIVOT operator determines how many columns are required to hold transpose and it can be combined with GROUP BY clause.

```
SELECT L1,..,Lj
,sum(CASE WHEN R1 = v11 and .. and Rk = vk1
THEN A ELSE null END)

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

Listing 1 – Shows optimized instructions for PIVOT construct

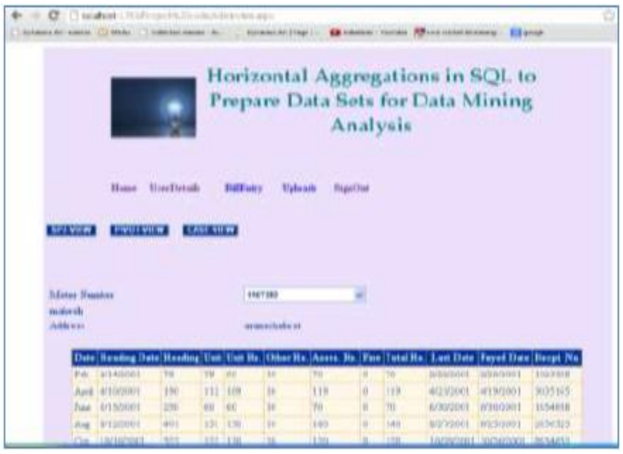
As can be seen in listing1, the optimized query projects only the columns that participate in computation of horizontal aggregations.

3 CASE Method

This construct is built based on the existing CASE struct of SQL. Based on Boolean expression one of the results is returned by CASE construct. It is same as projection/aggregation query from relational point of view. Based on some conjunction of conditions each non key value is given by a function. Here two basic strategies to compute horizontal aggregations. The first strategy is to compute directly from input table. The second approach is to compute vertical aggregation and save the results into temporary table. Then that table is further used to compute horizontal aggregations. The actual implementation is based on the details given in [18].

V. EXPERIMENTAL EVALUATION

The environment used for the development of prototype web based application that demonstrates the three horizontal aggregations include Visual Studio 2010, and SQL Server 2008. The former is used to build front end application with web based interface while the latter is used to store data permanently. The technologies used include ASP.NET which is meant for developing web services and web applications, and AJAX (Asynchronous JavaScript and XML) for rich user experience. Programming language used in C# which is an object oriented high level programming language. The result of horizontal aggregation SPJ is shown in fig.4.



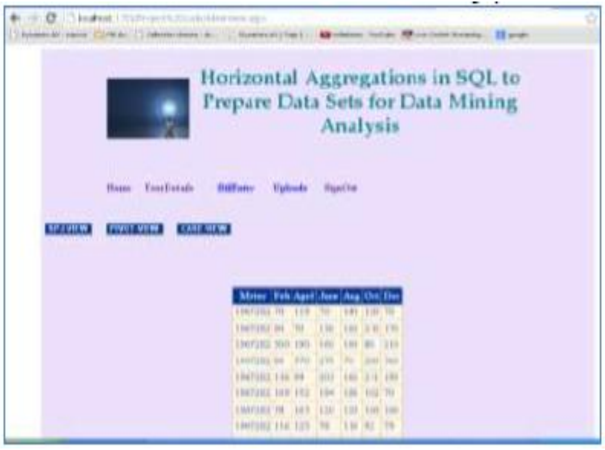
Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis

Meter Number 1987380
mahesh
Address [illegible]

Date	Reading Date	Reading	Unit	Unit Rs.	Other Rs.	Assess. Rs.	Fine	Total Rs.	Last Date	Payed Date	Recpt. No
Feb	2/14/2001	78	78	60	10	70	0	70	[illegible]	[illegible]	[illegible]
April	4/10/2001	190	111	109	10	119	0	119	4/23/2001	4/19/2001	303165
June	6/15/2001	250	60	60	10	70	0	70	6/30/2001	6/30/2001	1554858
Aug	8/12/2001	401	131	130	10	140	0	140	8/27/2001	8/25/2001	2656325
Oct	10/16/2001	[illegible]	[illegible]	120	10	130	0	130	10/29/2001	10/26/2001	[illegible]

Fig. 4 – Results of SJP aggregation

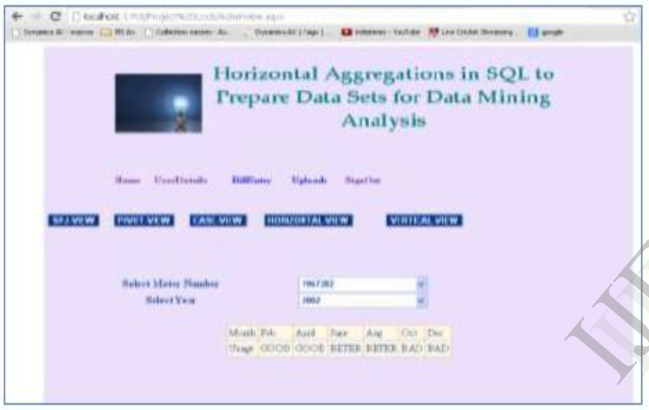
As can be seen in fig. 4, the results are through the SPJ operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.


Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis
Home
UserDetails
BillRates
Uploads
SignOut
SPJ VIEW
PIVOT VIEW
CASE VIEW

Meter	Feb	April	June	Aug	Oct	Dec
1967202	70	119	70	149	110	70
1967202	84	70	130	140	210	170
1967202	300	190	140	190	80	210
1967202	84	170	270	70	200	160
1967202	116	89	203	140	210	180
1967202	169	152	194	120	102	70
1967202	78	167	120	120	110	100
1967202	116	123	70	130	82	78

Fig. 5 – Result of Pivoting Aggregation

As can be seen in fig. 5, the results are through the PIVOT operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.



Select Meter Number: 1967202
Select Year: 2002

Month	Feb	April	June	Aug	Oct	Dec
Usage	GOOD	GOOD	BETER	BETER	BAD	BAD

Fig. 6 – Result of CASE Aggregation

As can be seen in fig. 6, the results are through the CASE operation that results in data in horizontal layout. Data in this layout can be considered as data set that can be used for further data mining operations.

VI. CONCLUSION

In this paper I have extended three aggregate functions such as CASE, SPJ and PIVOT. These are known as horizontal aggregations. I 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. Instead of single value, the horizontal aggregations return a set of values in the form of a row. The result resembles a multidimensional vector. I have implemented SPJ using standard relational query operations. The CASE construct is developed extending SQL CASE. The PIVOT makes use of built in operator provided by RDBMS for pivoting data. To evaluate these operators, we have developed a web based prototype application and results reveal that the proposed horizontal aggregations are capable of preparing data sets for real world data mining operations.

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] G. Bhargava, P. Goel, and B.R. Iyer, "Hypergraph Based Reorderings of Outer Join Queries with Complex Predicates," Proc. ACMSIGMOD Int'l Conf. Management of Data (SIGMOD '95), pp. 304-315, 1995.
[12] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group -by, Cross-Tab and Sub-Total," Proc. Int'l Conf. Data Eng., pp. 152-159, 1996.
[13] G. Graefe, U. Fayyad, and S. Chaudhuri, "On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases," Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD '98), pp. 204-208, 1998.
[14] 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.
[15] C. Cunningham, G. Graefe, and C.A. Galindo-Legaria, "PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS," Proc. 13th Int'l Conf. Very Large Data Bases (VLDB '04), pp. 998-1009, 2004.
[16] C. Ordonez, "Horizontal Aggregations for Building Tabular Data Sets," Proc. Ninth ACM SIGMOD Workshop Data Mining and Knowledge Discovery (DMKD '04), pp. 35-42, 2004.
[17] C. Ordonez, "Vertical and Horizontal Percentage Aggregations," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '04), pp. 866-871, 2004.
[18] Carlos Ordonez and Zhibo Chen, "Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 4, APRIL 2012.