International Journal for Research in Emerging Science and Technology, Volume-1, Issue-2, July 2014, ISSN: 2349-7610

Horizontal Aggregations Based Data Sets for Data Mining Analysis: A Review

[1]Mr.Gaurav J.Sawale and [2] Prof. Dr.S. R.Gupta
[1]Department of computer science & engineering, PRMIT&R, Badnera, Amravati, Maharashtra, India
[2]Department of computer science & engineering, PRMIT&R, Badnera, Amravati, Maharashtra, India

Abstract

Preparing a data set for analysis is generally 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. In general, a significant manual effort is required to build data sets, where a horizontal layout is required. We propose simple, yet powerful, methods to generate SQL code to return aggregated columns in a horizontal tabular layout, returning a set of numbers instead of one number per row. This new class of functions is called horizontal aggregations. Horizontal aggregations build data sets with a horizontal denormalized layout (e.g. point-dimension, observation-variable, instance-feature), which is the standard layout required by most data mining algorithms. We propose three fundamental methods to evaluate horizontal aggregations: CASE: Exploiting the programming CASE construct; SPJ: Based on standard relational algebra operators (SPJ queries); PIVOT: Using the PIVOT operator, which is offered by some DBMSs. Experiments with large tables compare the proposed query evaluation methods. Our CASE method has similar speed to the PIVOT operator and it is much faster than the SPJ method. In general, the CASE and PIVOT methods exhibit linear scalability, whereas the SPJ method does not.

Keyword: Aggregation; Data Preparation; Pivoting; SQL

I. Introduction

In a relational database, especially with normalized tables, a significant effort is required to prepare a summary data set [16] that can be used as input for a data mining or statistical algorithm [17] [15]. Most algorithms require as input a data set with a horizontal layout, with several records and one variable or dimension per column. Consider the case with models like clustering, classification, regression, and PCA; consult [10] [15]. 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 report 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. A show evaluating horizontal aggregations is a challenging and interesting problem and A introduce alternative methods and optimizations for their efficient evaluation. Horizontal aggregation is new class of function to return aggregated columns in a horizontal layout. Most algorithms require datasets with horizontal layout as input with several records and one variable or dimensions per columns. Managing large data sets without DBMS support can be a difficult task. Trying different subsets of data points and dimensions is more flexible, faster and easier to do inside a relational database with SQL queries than outside with alternative tool. Horizontal aggregations can be performing by using operator, it can easily be implemented inside a query processor, much like a select, project and join. PIVOT operator on tabular data that exchange rows, enable data analysis, and data presentation. There are many existing functions and operators for aggregation in SQL. The most commonly used aggregations is the sum of a column And other aggregation operators return the average, maximum, minimum or row count over groups of rows. All operations for aggregation have many limitations to

build large data sets for data mining purposes. Database schemas are also highly normalized for On-Line Transaction Processing (OLTP) systems where data sets that are stored in a relational database or data warehouse. But data mining, statistical or machine learning algorithms generally require aggregated data in summarized form. Data mining algorithm requires suitable input in the form of cross tabular (horizontal) form, significant effort is required to compute aggregations for this purpose. Such effort is due to the amount and complexity of SQL code which needs to be written, optimized and tested. Data aggregation is a process in which information is gathered and expressed in a summary form, and which is used for purposes such as statistical analysis. A common aggregation purpose is to get more information about particular groups based on specific variables such as age, name, phone number, address, profession, or income. Most algorithms require input as a data set with a horizontal layout, with several records and one variable or dimension per column. That technique is used with models like clustering, classification, regression and PCA. Dimension used in data mining technique are point dimension.

1.1 Motivation

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 [16]. The most widely known aggregation is the sum of a column over groups of rows. Some other aggregations 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 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 Online Transaction Processing (OLTP) systems where database schemas are highly normalized. 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 crosstabular (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

1.2 Objectives

1) The proposed dissertation objective is a new class of extended aggregate functions, called horizontal aggregations which helps in investigating data sets for data mining and OLAP cube exploration.
2) It also investigate three query evaluation method for horizontal aggregation in order to optimize the query required for preparing summary data sets and the three methods are SPJ, PIVOT and CASE.
3) Combining any of three methods(SPJ, PIVOT , CASE) with horizontal aggregations used to generate SQL code to build data set for data mining. Aim is to provide a more efficient, better integrated and more secure solution compared to external data mining tools.

II. Literature Review

2.1 Data Mining

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. 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. The Scope of Data Mining Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and

quality, data mining technology can generate new business opportunities by providing these capabilities:

Automated prediction of trends and behaviors: Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

Automated discovery of previously unknown patterns: Data mining tools sweep through databases and identify previously hidden patterns in one step.The most commonly used techniques in data mining are:

- Artificial neural networks: Non-linear predictive models that learn through training and resemble biological neural networks in structure.
- Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID).
- Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.
- Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset sometimes called the k-nearest neighbor technique.
- Rule induction: The extraction of useful if-then rules from data based on statistical significance.

2.2 Architecture for Data Mining

Data mining is an iterative process that typically involve following phases:

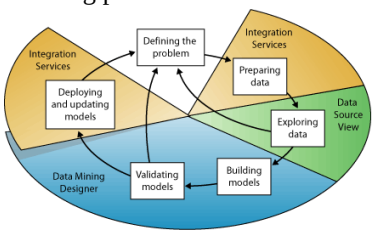

Fig. 1. Data mining architecture

Problem definition: A data mining project starts with the understanding of the business problem. Data mining experts, business experts, and domain experts work closely together to define the project objectives and the requirements from a business perspective. The project objective is then translated into a data mining problem definition. In the problem definition phase, data mining tools are not yet required [1].

Data exploration: Domain experts understand the meaning of the metadata. They collect, describe, and explore the data. They also identify quality problems of the data. A frequent exchange with the data mining experts and the business experts from the problem definition phase is vital. In the data exploration phase, traditional data analysis tools, for example, statistics, are used to explore the data [1].

Data preparation: Domain experts build the data model for the modeling process. They collect, cleanse, and format the data because some of the mining functions accept data only in a certain format. They also create new derived attributes, for example, an average value. In the data preparation phase, data is tweaked multiple times in no prescribed order. Preparing the data for the modeling tool by selecting tables, records, and attributes, are typical tasks in this phase. The meaning of the data is not changed [1].

Modeling: Data mining experts select and apply various mining functions because you can use different mining functions for the same type of data mining problem. Some of the mining functions require specific data types. The data mining experts must assess each model. In the modeling phase, a frequent exchange with the domain experts from the data preparation phase is required. The modeling phase and the evaluation phase are coupled. They can be repeated several times to change parameters until optimal values are achieved. When the final

modeling phase is completed, a model of high quality has been built [1].

Evaluation: Data mining experts evaluate the model. If the model does not satisfy their expectations, they go back to the modeling phase and rebuild the model by changing its parameters until optimal values are achieved.

2.4 Role of SQL in Data Mining

SQL propose three methods to evaluate horizontal aggregations.The first method relies only on relational operations.That is, only doing select, project, join, and aggregation queries; A call it the SPJ method. The second form relies on the SQL "CASE" construct; A call it the CASE method. Each table has an index on its primary key for efficient join processing. The third method uses the built-in PIVOT operator, which transforms rows to columns (e.g., transposing).

2.4.1 SPJ Method

The SPJ method is interesting from a theoretical point of view because it is based on relational operators only. The basic idea is to create one table with a vertical aggregation for each result column, and then join all those tables to produce FH. A aggregate from F into d projected tables with d Select-Project-Join-Aggregation queries (selection, projection, join, aggregation). Each table FI corresponds to one subgrouping combination and has {L1, . . .Lj} as primary key and an aggregation on A as the only nonkey column. It is necessary to introduce an additional table F0 that will be outer joined with projected tables to get a complete result set. A propose two basic substrategies to compute FH. The first one directly aggregates from F. The second one computes the equivalent vertical aggregation in a temporary table FV grouping by L1; . . . ; Lj; R1; . . .;Rk. Then horizontal aggregations can be instead computed from FV, which is a compressed version of F, since standard aggregations are distributive [9]. The SPJ method code is as follows (computed from F):

```
INSERT INTO F1
SELECT D1,sum(A) AS A
FROM F
WHERE D2='X'
GROUP BY D1;
INSERT INTO F2
SELECT D1,sum(A) AS A
FROM F
WHERE D2='Y'
GROUP BY D1;
INSERT INTO FH
SELECT F0.D1,F1.A AS D2_X,F2.A AS D2_Y
FROM F0 LEFT OUTER JOIN F1 on F0.D1=F1.D1
LEFT OUTER JOIN F2 on F0.D1=F2.D1;
```

2.4.2 CASE Method

For this method, A use the "case" programming construct available in SQL. The case statement returns a value selected from a set of values based on boolean expressions.From a relational database theory point of view this is equivalent to doing a simple projection/aggregation query where each nonkey value is given by a function that returns a number based on some conjunction of conditions. The two basic substrategies to compute FH. In a similar manner to SPJ, the first one directly aggregates from F and the second one computes the vertical aggregation in a temporary table FV and then horizontal aggregations are indirectly computed from FV.A now present the direct aggregation method. Horizontal aggregation queries can be evaluated by directly aggregating from F and transposing rows at the same time to produce FH. First, there is need to get the unique combinations of R1; . . .;Rk that define the matching boolean expression for result columns. The SQL code to compute horizontal aggregations directly from F is as follows:

V () is a standard (vertical) SQL aggregation that has a "case" statement as argument. Horizontal aggregations need to set the result to null when there are no qualifying rows for the specific horizontal group to be consistent with the SPJ method and also with the extended relational model [4]. The CASE method code is as follows (computed from F):

```
INSERT INTO FH
SELECT D1,
SUM(CASE WHEN D2='X' THEN A
ELSE null END) as D2_X,
SUM(CASE WHEN D2='Y' THEN A
 ELSE null END) as D2_Y
FROM F
GROUP BY D1;
```

2.4.3 PIVOT Method

Consider the PIVOT operator which is a built-in operator in a commercial DBMS. Since this operator can perform transposition it can help evaluating horizontal aggregations.The PIVOT method internally needs to determine how many columns are needed to store the transposed table and it can be combined with the GROUP BY clause.The PIVOT method SQL is as follows (computed from F):

```
INSERT INTO FH
SELECT D1,
[X] as D2_X
[Y] as D2_Y
FROM (SELECT D1, D2, A FROM F) as p
PIVOT (
SUM(A)
FOR D2 IN ([X], [Y]))
AS PIVOT;
```

2.5 Summary and Discussion

For all proposed methods to evaluate horizontal aggregations a summarize common requirements are as follows:

All methods require grouping rows by L1; . . . ; Lj in one or several queries.

a) All methods must initially get all distinct combinations of R1; . . .;Rk to know the number and names of result columns. Each combination will match an input row with a result column. This step makes query optimization difficult by standard query optimization methods because such columns cannot be known when a horizontal aggregation query is parsed and optimized.

b) It is necessary to set result columns to null when there are no qualifying rows. This is done either by outer joins or by the CASE statement.

c) Computation can be accelerated in some cases by first computing FV and then computing further aggregations from FV instead of F. The amount of acceleration depends on how larger is N with respect to n (i.e., if N _ n). These requirements can be used to develop more efficient query evaluation algorithms.

III System Analysis and Design

3.1 Horizontal Aggregations

As proposed a new class of aggregations that have similar behavior to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, A 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 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. A start by explaining how to automatically generate SQL code.

F

K	D_1	D_2	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

F_V

D_1	D_2	A
1	X	null
1	Y	10
2	X	8
2	Y	6
3	X	17

F_H

D_1	D_2X	D_2Y
1	null	10
2	8	6
3	17	null

Fig. 2. Example of F, FV, and FH

A Traditional vertical sum() aggregation stored in FV , and a horizontal aggregation stored in FH. The basic SQL aggregation query is:

```
SELECT D1;D2, sum(A)
FROM  F
GROUP BY D1, D2
ORDER BY D1, D2;
```

The table shows FV has only five rows because D1= 3 and D2 =Y do not appear together. Also, the first row in FV has null in A following SQL evaluation semantics. On the other hand, table FH has three rows and two (d =2) nonkey columns, effectively storing six aggregated values. In FH it is necessary to populate the last row with null. Therefore, nulls may come from F or may be introduced by the horizontal layout.

3.2 Existing System

An existing to preparing a data set for analysis is generally 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.

Disadvantage:

Existing SQL aggregations have limitations to prepare data sets.

To return one column per aggregated group

3.2.1 Previous Process Flow:

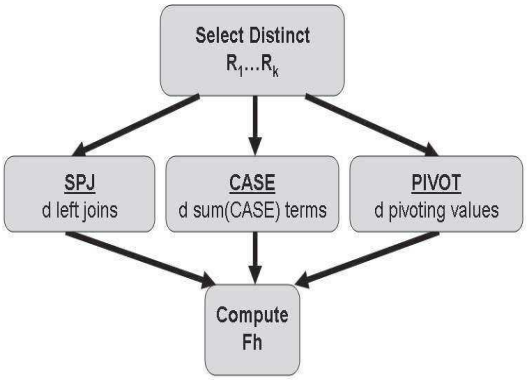

Fig 3: Previous Process Flow

3.3 Proposed System

Proposed horizontal aggregations provide several unique features and advantages. First, they represent a template to generate SQL code from a data mining tool. Such SQL code automates writing SQL queries, optimizing them and testing them for correctness.

Advantage:

1) The SQL code reduces manual work in the data preparation phase in a data mining project.

2) The SQL code is automatically generated it is likely to be more efficient than SQL code written by an end user.

3) The data sets can be created in less time.

4) The data set can be created entirely inside the DBMS

3.3.1 Proposed Process Flow:

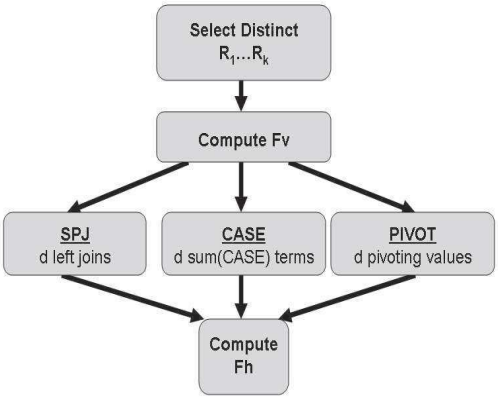

Fig 4: Proposed Process Flow

Proposed System Components:

1. Admin Module
2. User Module
3. View Module
4. Download Module

Module 1 : Admin Module

Admin will upload new connection form based on regulations in various states. Admin will be able to 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.

Module 2 : 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 alloAd to change or edit and save it.

Module 3 : View Module

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

i) SPJ

ii) PIVOT

iii) CASE

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

ii) 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.

iii) CASE :

Using CASE query A 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 vieAd in two different 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.

Module 4:

Download Module User will be able to download the various details regarding bills. If he/she is a new user, he/she can download the new connection form, subscription details etc. then he/she can download his /her previous bill details in hands so as to ensure it.

IV. Conclusion

A new class of extended aggregate functions, called horizontal aggregations which help 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 multi-dimensional vector. 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 our most important contribution. Our 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. The fact that the output horizontal columns are not available when the query is plan is explored and chosen to make it evaluated through standard SQL mechanisms infeasibly.Experiments with large tables show our proposed horizontal aggregations evaluated with the CASE method have similar performance to the built-in PIVOT operator. A believe that the 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

V. References

[1] G. Bhargava, P. Goel, and B.R. Iyer. Hypergraph based reorderings of outer join queries with complex predicates. In *ACM SIGMOD Conference*, pages 304–315, 1995.

[2] J.A. Blakeley, V. Rao, I. Kunen, A. Prout, M. Henaire, and C. Klein-erman. .NET database programmability and extensibility in Microsoft SQL Server. In *Proc. ACM SIGMOD Conference*, pages 1087–1098, 2008.

[3] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman. Non-stop SQL/MX primitives for knowledge discovery. In *ACM KDD Conference*, pages 425–429, 1999.

[4] E.F. Codd. Extending the database relational model to capture more meaning. *ACM TODS*, 4(4):397–434, 1979.

[5] 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.

[6] C. Galindo-Legaria and A. Rosenthal. Outer join simplification and reordering for query optimization. *ACM TODS*, 22(1):43–73, 1997.

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

[8] G. Graefe, U. Fayyad, and S. Chaudhuri. On the efficient gathering of sufficient statistics for classification from large SQL databases. In *Proc. ACM KDD Conference*, pages 204–208, 1998.

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

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

[11] G. Luo, J.F. Naughton, C.J. Ellmann, and M. Watzke. Locking protocols for materialized aggregate join views. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6):796–807, 2005.

[12] C. Ordonez. Horizontal aggregations for building tabular data sets. In *Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop*, pages 35–42, 2004.

[13] C. Ordonez. Vertical and horizontal percentage

aggregations. In *Proc. ACM SIGMOD Conference*, pages 866–871, 2004.

[14] C. Ordonez. Integrating K-means clustering with a relational DBMS using SQL. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(2):188–201, 2006.

[15] C. Ordonez. Statistical model computation with UDFs. *IEEE Transac-tions on Knowledge and Data Engineering (TKDE)*, 22, 2010.

[16] C. Ordonez. Data set preprocessing and transformation in a database system.

[17] C. Ordonez and S. Pitchaimalai. Bayesian classifiers programmed in SQL. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(1):139–144, 2010.

[18] 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.

[19] H. Wang, C. Zaniolo, and C.R. Luo. ATLaS: A small but complete SQL extension for data mining and data streams. In *Proc. VLDB Conference*, pages 1113–1116, 2003.

[20] 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.