Efficient Tabular Dataset Preparations by the Aggregations in SQL: A Survey

Jincy Annie V.V
Post Graduate Student
Karunya university
Coimbatore, India

J. A. M. Rexie
Assistant Professor
Karunya University
Coimbatore, India

ABSTRACT

Many data mining algorithms require the result to be transformed into tabular format. Tabular datasets are the suitable input for many data mining approaches. But the existing SQL aggregations cannot produce results in tabular form with more summarized details especially in horizontal tabular form. Here discuss several approaches to produce data sets in tabular format and also present an efficient method to produce results in horizontal tabular format. Alternative methods for the evaluation of new format are also shown here.

Keywords

Aggregations, Vertical Aggregations, Horizontal Aggregations, Structured Query Language, Dataset Preparation

1. INTRODUCTION

Data mining is used for the extraction of useful information from large databases. The datasets can be transformed to understandable format for further uses. This involves database and data management aspects, data preprocessing, model and interface considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.

Tabular datasets are the required input for many data mining algorithms and statistical algorithms [1], [2]. To produce data in tabular summarized form, a relational database requires large efforts. A relational database system produce normalized tables for analysis. To get more details for analysis, denormalized tables are used. With the help of SQL queries users can perform aggregation of tables and can produce results in vertical and horizontal layout.

2. AGGREGATIONS

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 ( ). Data aggregation can be user-based [3]. For example personal data aggregation services offer the user a single point for collection of their personal information from other Web sites.

Vertical aggregations: Vertical aggregation is similar to standard SQL aggregations. This produces results in a vertical format and contains more rows. There are some approaches which produce results in vertically aggregated form.

Horizontal aggregations: Horizontal aggregations are also similar to standard SQL aggregations but this can produce results in horizontal tabular format. Here data sets for all the operations are produced from some data mining tool and apply the aggregation operations on that dataset. To produce results in horizontal layout small syntax extensions to normal SQL syntax is needed. The syntax for horizontal aggregation is given below.

SELECT columns, Aggregation Function

(Measure column BY Aggregating Parameters)

FROM GROUPING columns

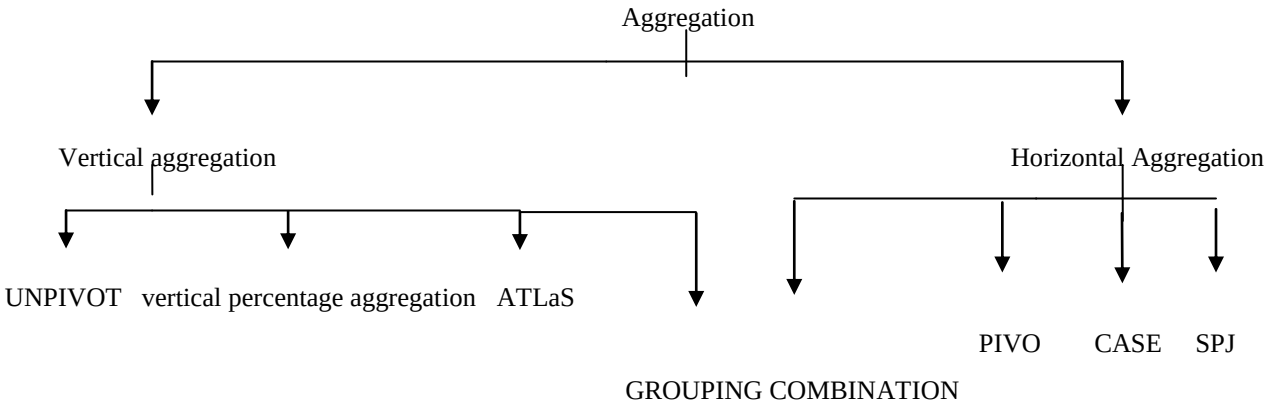

Fig1.Aggregation classification

3. DATASET PREPARATION

Dataset preparation [4] 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 denormalized table than normalized table. Because that contain more details than normalized tables and many analysis require analysis on large amount of data.

The major problems in the creation and transformation of variables for analysis are selection of appropriate record from the available large dataset and preparation of efficient SQL queries to optimize them. Most the issues in the creation and transformation of dataset is related to summarization, aggregation, denormalization, cross-tabulation. Sometimes analysis need summarized details then detailed. So there is a need for summarized data. Cross-tabulation is also an important concept because it gives detailed information after analysis in a horizontal tabular format. Those are easy to understand than vertical format. Horizontal aggregation almost similar to normal aggregation but it uses some syntax extensions. Selecting SQL queries face some difficulties when using left outer join for effectiveness and use of appropriate SQL queries for each operation and handling of multiple primary keys.

There is a data preparation framework [5] for efficiently preparing dataset for analysis. Quality of any analysis depends on the quality of data being processed. Since dataset preparation is most expensive and time consuming task, this dataset preparation is very important. The data is stored inside the database system can get the benefit of database management system. There are four steps for the dataset preparation. Dataset preparation start 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.

4. STRUCTURED QUERY LANGUAGE IN DATA MINING

Data mining require the extraction of information from large database. So programmers can take the advantage of relational database system. There are several data mining approaches that use SQL extensions for information extraction. For example, integration of association rule uses SQL implementations with the support of an apriory algorithm [6]. But the use of normal SQL queries has low performance than other architectures. So every approaches use some extensions on the normal SQL syntax to improve the performance. K-means clustering also use SQL implementations to get the benefit of DBMS [7]. There is another approach, Bayesian classifier also use SQL for operations. The major advantages of using DBMS are storage management, concurrency control, and security.

5. COMPARATIVE STUDY

In data mining many approaches need tabular datasets for operations. Tabular datasets are easier to understand and use than any other approaches. Aggregations using SQL play a major role in producing tabular datasets. Here various approaches for performing aggregation are taken for comparison.

5.1 GROUPING COMBINATION

GROUPING COMBINATION operator [8] is developed to handle the aggregation and grouping of high dimensional data. This operator can solve limitations of normal GROUPING operator. The operators like GROUPING SET, ROLLUP, and CUBE can also perform aggregation and can produce tabular results. But these are difficult to use when the available input dataset is very large. When the available input dataset is large GROUPING SET operator require long complex SQL queries. The ROLL UP operator can perform aggregation on smaller datasets and produce tabular results vertical format. But the vertical format is not efficient for many data mining approaches. The CUBE operator can perform aggregations on large datasets. But the CUBE operator eliminates some of the details when aggregation is performed. Because of these limitations GROUPING COMBINATION operator is developed. But the GROUPING COMBINATION operator can implemented only with the help of complex algorithms. So its performance is low in the case of execution.

5.2 ATLaS

ATLaS is a database language developed to solve the limitations of SQL operator. ATLaS [9] can perform aggregations that are not possible with standard SQL. Standard SQL can support only basic aggregation operations. This language use aggregations and table functions in SQL. To perform operations in ATLAS entire SQL statement is divided into three functions INTIALIZE, ITERATE, TERMINATE. Declarations are given in the INITIALIZE section. The major operation is specified in the ITERATIVE section. The final statement to execute is specified in the TERMINATE sections. The major advantage of ATLaS is that it can support online aggregations. In online aggregation user evaluate aggregation query in an online fashion execution. But the execution of ATLaS operator consumes more space than executing with normal SQL. Also it cannot results in horizontal tabular format.

5.3 Vertical and Horizontal percentage aggregations

This aggregations help to calculate percentages for operations using vertical and horizontal aggregations [10]. Vertical percentage aggregation returns one row for percentage in vertical format. Horizontal percentage aggregation returns entire 100% of results on the same row. This percentage aggregation used only for computing percentages in vertical or horizontal format. These aggregations are similar to normal vertical and horizontal aggregation except that it can compute results only in percentage format. So there may be extra work in the percentage conversion when other computations are required on the dataset.

5.4 UNPIVOT Operator

UNPIVOT operator is also an aggregation operator for producing results in tabular format. This operator works in

opposite of PIVOT operator that is they transform columns into rows. This creates additional rows from columns to produce a big table. Because of these vertical layout it cannot used for most of the mining algorithms which require horizontal table as input. UNPIVOT operator [11] is commonly used for the statistical computation of some data mining approaches. The normal syntax is given below.

SELECT columns FROM table UNPIVOT

(Measure Column FOR Pivot Column IN (Pivot Column Values))

5.5 Interpreted Storage Format

This is developed to handle null values in horizontal and vertical layouts. Interpreted format can handle all the sparse data management complexities [12]. Horizontal aggregation requires more space due to large number of null values. Vertical aggregations have small number of null values. Interpreted storage format store nothing for null attributes. When the tuple has value for an attribute in the table, attribute identifier (attribute_id), a length field, value appears in the tuple. This stored along with particular head. The major problem here is that the value stored in this format is not easily accessible for operations.

5.6 PIVOT Operator

The PIVOT operator is built in operator in commercial DBMS. PIVOT operator [11], [13] is used with standard select statement by using small syntax extensions. This operator transforms rows into columns to produce horizontal layout. Performance of PIVOT operator is high compared with other operators. PIVOT operator performs well even though the dataset is very large. The major advantage of PIVOT operator is that it can solve the upper limit limitation of DBMS. The basic syntax for PIVOT method is given below.

SELECT columns FROM table PIVOT (
Aggregate Function(Measure Column) FOR Pivot Column IN ([Pivot Column Values]))AS Alias

5.4 SPJ Method

In SPJ method [13] create one table with a vertical aggregation for each result column, and then join all those tables top produce FH. Here d projected tables are created from d select, project, join queries. Left outer join queries are used to join all the projected tables. SPJ method can produce tables in horizontal layout an optimized SPJ method can produce more efficient result. The performance of SPJ approach is very low when there is large number of rows. But this can perform aggregation with the help of basic SQL queries. This is easier to support by any database.

CASE Method: CASE method can be performed by combining GROUP-BY and CASE statements [13]. 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.

SELECT columns, Aggregate Function

(CASE WHEN Boolean expression THEN result ELSE result expression END)

FROM table GROUP BY columns

Table 1

Comparison of aggregation approaches

Method Discussed	Type of Aggregation	Feature
GROUPING COMBINATION Operator	Horizontal and Vertical	Implemented with complex algorithms
ATLaS	Vertical Aggregation	Solve limitation of normal SQL
Vertical and Horizontal percentage aggregations	Vertical and horizontal aggregation	Can only operate on percentages
Interpreted Storage Format	Vertical and horizontal aggregation	Data retrieval is difficult
UNPIVOT Operator	Vertical aggregation	Give results in vertical layout
PIVOT Operator	Horizontal aggregation	Use small syntax extension in select statement
SPJ Method	Horizontal aggregation	Use select, project and join queries
CASE Method	Horizontal aggregation	Use small syntax extensions to select statement

6. CONCLUSION

Here in this paper various approaches for performing aggregation are presented. Among this most data mining algorithms require the methods which produce horizontal tabular datasets. This is because in horizontal layout each row contains more details instead of one number in a row. It produce resultant table with more column and few rows. Here there are three methods which produce horizontal table. This paper takes the advantage all the three methods in aggregation and uses them for data mining approaches. The three methods are PIVOT, CASE and SPJ. PIVOT and CASE use small syntax extensions to the normal select statement. SPJ is the combination of normal select, project and join methods.

7. REFERENCES

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

[2] 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.

[3] Haixun Wang, Carlo Zaniolo, "User Defined Aggregates in Object-Relational Systems," *IEEE Trans. Knowledge and Data Eng*., 2001.

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

[5] Kai-Uwe Sattler, Eike Schallehn, "A Data Preparation Framework based on a Multidatabase Language**,**" *IEEE Trans. Knowledge and Data Eng*, 2001.

[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] 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.

[8] Alexander Hinneburg, Dirk Wolfgang Lehner, "Combi-Operator-Database Support for Data Mining Applications," *Proc. 29th VLDB Conference*, 2003.

[9] 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.

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

[11] 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.

[12] Jennifer L. Beckmann, Alan Halverson, Rajasekar Krishnamurthy, Jeffrey F. Naughton, "Extending RDBMSs to Support Sparse Datasets Using An Interpreted Attribute Storage Format," *An enterprise directory solution with DB2. IBM Systems Journal*, 39(2), 2005.

[13] C. Ordonez, "Horizontal Aggregations for Building Tabular Data Sets," *IEEE Trans. Knowledge and Data Eng*, VOL. 24, NO. 4, April 2012.