Volume 3, Issue 3, March 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Efficient Approach for Building Dataset in Data Mining

Durka.C
Computer Science and Engineering
Bharath Institute of Science and Technology
Chennai, India

Kerana Hanirex.D
Computer Science and Engineering
Bharath Institute of Science and Technology
Chennai, India

Abstract--Data Mining is one of the emerging field in Research. Preparing a dataset is one of the important task in data mining. To analyze data efficiently, Data mining systems are widely using datasets with columns in horizontal tabular layout. In normal, a significant manual effort is required to build data sets, where a horizontal tabular layout. Building a datasets for analysis is normally a most time consuming task. Existing SQL aggregations have limitation to build data sets because they return one column for aggregated group using group functions. 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.It does mean that this paper achieves horizontal aggregations throughsome constructs built that includes SQL queries as well. In the proposed system, a new standard of pivoting option is incorporated using Data mining. This can be achieved with the tool SAAS (SQL Server Analysis Services). Data will be taken and transformed into knowledge cubes. This can be achieved with MDX (Multi-DimensionaleXpression) queries. On top of that, the knowledge data will be customized based on "Generalized and Suppression Algorithm" .Its provide privacy for dataset.

Keywords: Aggregation, Data Preparation, Generalized and suppression, SQL.

INTRODUCTION

Preparing a database needs identification of relevant data and then normalizing the tables. Building the relevant data for data mining, is a most time consuming task. This task generally requires writing long SQL statements or customizing SQL code is needed if it is automatically generated by some tool. There are two main ingredients in such SQL code namely join and aggregation. The most widely-known aggregation is the sum of a column over group of rows. There are 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 On-Line Transaction Processing (OLTP) systems where database schemas are highly normalized. Based on current available functions and clauses in SQL, a significant effort is required to compute aggregations. Such effort is due to the size and complexity of SQL code which needs to be written, optimized and tested. In general aggregations are hard to interpret when there are many result rows. In order to perform the analysis of exported tables in spreadsheets it will be more convenient to have aggregations of the same group in one row. With such limitations in mind, we propose a new class of aggregate functions that aggregates numeric expressions and transpose results to produce a data set with a horizontal layout.

Functions belonging to this class are called horizontal aggregations. Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout instead of a single value per row. Horizontal aggregations provide several unique features and advantages. First, they represent a template to generate SQL code from a data mining tool. This SQL code reduces manual work in the data preparation phase in a data mining project. Second, since SQL code is automatically generated it is likely to be more efficient than SQL code written by an end user. Third, the data set can be created entirely inside the DBMS. 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.

Horizontal aggregation is evaluated by the following approach. *Case:* This method uses the "case" programming construct available in SQL. The case statement returns a value selected from a set of values based on Boolean expressions. *SPJ*: It 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 vertical aggregation for each result column, and then join all those tables to produce another table. [1] *Pivot*: It is a built-in operator offered by some DBMS, which transforms row to columns. This 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. 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. To overcome this problem, multidimensional data cube is used. It is also called as OLAP cubes. [13] A data cube is a collection of data that's been aggregated to allow queries to return data quickly. Transforming normal data into knowledge cube is one of the emerging fields in the current market. The data will be taken and it will be transformed into knowledge cubes. The data cube is created using the tool

SAAS (SQL Server Analysis Services). Microsoft SQL Server OLAP Services provides architecture for access to multidimensional data.

Generalized and Suppression: It is mainly used to provide privacy for datasets in data mining. Generalization consists of substituting attribute values with semantically consistent but less precise values. For example, the month of birth can be replaced by the year of birth which occurs in more records, so that the identification of a specific individual is more difficult. Generalization maintains the correctness of the data at the record level. Different systems use various methods for selecting the attributes and records for generalization as well as the generalization technique. Suppression refers to removing a certain attribute value and replacing occurrences of the value with a special Value like "?","*", indicating that any value can be placed instead. Suppression can drastically reduce the quality of the data. The paper is organized as follows: The reviews of the Related Work are presented in the Section 2. Section 3 describes Method definition, Section 4 describes the Experimental Analysis and Result and Conclusion and Future work are described in Section 5.

RELATED WORK

SQL extensions to define aggregate functions for association rule mining. Their optimizations have the purpose of avoiding joins to express cell formulas, but are not optimized to perform partial transposition for each group of result rows. ConorCunningalam [1] proposed an optimization and execution strategies in an RDBMS which uses two operators i.e., PIVOT operator on tabular data that exchange rows and columns, enable data transformations useful in data modeling, data analysis, and data presentation. They can quite easily be implemented inside a query processor system, much like select, project, and join operator. Such a design provides opportunities for better performance, both during query optimization and query execution. Pivot is an extension of Group By with unique restrictions and optimization opportunities, and this makes it very easy to introduce incrementally on top of existing grouping implementations.

H Wang.C.Zaniolo [2] proposed a small but complete SQL Extension for data mining and data Streams. This technique is a powerful database language and system that enables users to develop complete data-intensive applications in SQL by writing new aggregates and table functions in SQL, rather than in procedural languages as in current Object-Relational systems. The ATLaS system consist of applications including various data mining functions, that have been coded in ATLaS" SQL, and execute with a modest (20–40%) performance overhead with respect to the same applications written in C/C++. This system can handle continuous queries using the schema and queries in Query Repository.

C. Ordonez (2004)[3] introduced two aggregation functions. The first function returns one row for each percentage in vertical form like standard SQL aggregations. The second function returns each set of percentages adding 100% on the same row in horizontal form. These novel aggregate functions are used as a framework to introduce the concept of percentage queries and to generate efficient SQL code. Experiments study different percentage query optimization strategies and compare evaluation time of percentage queries. The advantage is that horizontal aggregation reduces the number of rows and columns. Disadvantage is that vertical aggregation increase the number of rows and columns. Thus increases the complexity.

G. Luo and J.F. Naughton (2005) [4]developed the immediate materialized view maintenance with transaction consistency where is enforced by generic concurrency control mechanism. A latch pool for aggregate join view is introduced. The latches in the latch pool guarantee that for each aggregate group, at most one tuple corresponding to this group exists in the aggregate join view. The main advantage, is that deadlock problem is solved. The main disadvantage is many join operations are used.

J. Gray, A. Bosworth, A. Layman, and H. Pirahesh [5] proposed a relational aggregation operator that generalizing Group-By, Cross-Tab, and Sub-Totals. The cube operator generalizes the histogram, cross tabulation, roll-up, drill-down, and sub-total constructs. The cube operator can be imbedded in more complex non-procedural data analysis programs and data mining. The cube operator treats each of the N aggregation attributes as a dimension of N-space. The aggregate of a particular set of attribute values is a point in this space and the set of points forms an N-dimensional cube. Super-aggregates are computed by aggregating the N-cube to lower dimensional spaces. Creating a data cube requires generating the power set (set of all subsets) of the aggregation columns. Since the CUBE is an aggregation operation, it makes sense to externalize it by overloading the SQL GROUPBY operator.

METHOD DEFINITION

A new class of aggregation function called Horizontal aggregation, represents an extended form of traditional SQL (Structured Query Language) aggregation, which returns set of values in a horizontal layout. Horizontal aggregation is evaluated by the following methods as defined.

A. Case

It can be used in any statement or clause that allows a valid expression. The case statement returns a value selected from a set of values based on Boolean expression. The Boolean expression for each case statement has a conjunction of K equality comparisons. Query evaluation needs to combine the desired aggregation with "case" statement for each distinct combination of values of R1,……..,Rk.

The optimized **case** method code is as follows:

```
SELECT DISTINCT RI1
FROM F;
INSERT INTO FH SELECT LE1,LE2,....,LEj,
V(CASE WHEN RI1=v11 and . . . Rk=vk1 THEN A ELSE null END)
..
V(CASE WHEN RI1=v1n and . . . Rk=vkn THEN A ELSE null END)
FROM F GROUP BY LE1,LE2,. . .,LEj
```

B. SPJ

It is based on standard relational algebra operators (SPJ queries). The basic idea is to create one table with a vertical aggregation for each result column, and then join all those tables to produce another table. It is necessary to introduce an additional table F0 that will be outer joined with projected tables to get a complete result set.

The optimized **SPJ** method code is follows:

```
    INSERT INTO FH
    SELECT F0 .LE1 , F0. LE2 ,..., F0. LEj,
    F1.A, F2 .A,......,Fn .A
    FROM F0
    LEFT OUTER JOIN F1
  ON F0. LE1= F1. LE1 and. . . and F0. LEj= F1. LEj
    . . . .
    LEFT OUTER JOIN Fn
ON F0. LE1= Fn. LE1 and. . . and F0. LEj= Fn. LEj
```

C. Pivot

The pivot operator is a built-in operator which transforms row to columns. It internally needs to determine how many columns are needed to store the transposed table and it can be combined with the GROUP BY clause. Since this operator can perform transposition it can help in evaluating horizontal aggregation.

The optimized **PIVOT** method SQL is as follows:

```
SELECT * FROM (Bill Table PIVOT (SUM (Amount) for Month in (“Jan‟,‟Feb‟,‟Mar‟)
```

D. Knowledge Cube Generation

The data will be taken and it will be transformed into knowledge cubes. The data cube is created using the tool SAAS (SQL Server Analysis Services). Microsoft SQL Server Analysis Services is part of Microsoft SQL Server, a database management system. The data will be customized based on “Generalized & Suppression” algorithm. In this algorithm, only the authorized person can view the data. Microsoft SQL Server OLAP Services provides architecture for access to multidimensional data. Here, the fig 1 shown the transforming normal data into knowledge cube and fig 2 shows the multidimensional data cube.

a) Suppression-Based Protocol

Privacy can be preserved by simply suppressing all sensitive data before any disclosure or computation occurs. In suppression-based protocol, mask the original value with the special value. Suppressioninvolves not releasing a value at all for unauthorized users.

Given a database, we can suppress specific attributes in particular records as dictated by our privacy policy. For a partial suppression, an exact attribute value can be replaced with asterisk (e.g. 1000 to*****).

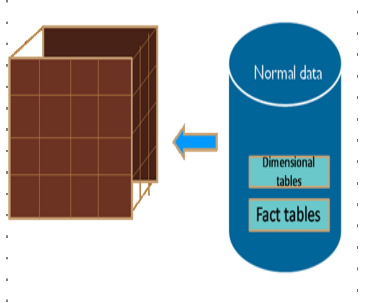

Fig1. Transforming data into knowledge cube

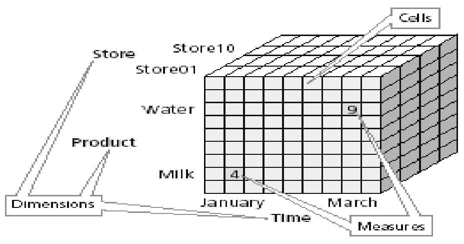

Fig2. Multidimensional Data Cube

b) Generalization-Based Protocol

In generalization-based protocol, the original values are replaced by more general ones.Generalizationinvolves replacing (or recoding) a value with a less specific but semantically consistent value.

Generalized can be applied at the level of single cell(substituting the cell value with a generalized version of it)or at the level of attribute generating all the cells in the corresponding column.

EXPERIMENTAL STUDY

For the experimental studies data base file from Database are taken. Attributes having integer values along with aggregate operator are chosen.

Administrator of the application can maintain the database and can upload new data. Administrator will upload new connection form based on the regulations. 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.

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 JavaScript and XML) for rich user experience. Programming language used in C# which is an object oriented high level programming language.

As next step, for attributes of horizontal aggregation i.e. having only integer values, a generalized and suppression algorithm is developed and thereafter privacy for particular attributes are developed.. The Table I shows the original data set. Table II& III Shown sample output of generalized and suppression algorithm.

TABLE I.
ORIGINAL DATA SET

Reading date	*Reading*	*Unit*	*Unit Rs*	*Total Rs*	*Receipt No*
12/12/2012	719	79	69	89	102356
05/02/2013	810	54	60	70	303516
15/04/2013	190	112	109	119	165478
08/06/2013	401	151	130	140	231456
14/08/2013	146	127	127	137	493465

TABLE II.
GENERALIZED DATA SET

Reading date	*Reading*	*Unit*	*Unit Rs*	*Total Rs*	*Receipt No*
12/12/2012	[0-1000]	79	69	89	[100000 -50000]
05/02/2013	[0-1000]	54	60	70	10000-50000
15/04/2013	[0-1000]	112	109	119	10000-50000
08/06/2013	[0-1000]	151	130	140	10000-50000
14/08/2013	[0-1000]	127	127	137	10000-50000

TABLE III.
SUPPRESSED DATA SET

Reading date	*Reading*	*Unit*	*Unit Rs*	*Total Rs*	*Receipt No*
12/12/2012	****	79	9	89	*******
05/02/2013	****	54	60	70	*******
15/04/2013	****	112	109	119	*******
08/06/2013	****	151	130	140	*******
14/08/2013	****	127	127	137	*******

Conclusion And Future Work

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 that includes SQL queries as well. The methods we use in 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. Many applications that employ data mining techniques involve mining data that include private and sensitive information about the subject. One way to enable effective data mining while preserving privacy is to anonymize the data set that includes private information about subject before being released for data mining. Most of the works are running behind analyzing the data and providing an estimated output.

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

Acknowledgment

The authors wished to thank the management of Bharath University for their renown able work and thinking.

References

[1] C. Cunningham, G. Graefe, and C.A. Galindo-Legeria, *"PIVOT AND UNPIVOT: Optimization and Execution Strategies in an RDBMS,"*Proc: 13th Int'l Conf. Very Large Data Bases (VLDS'04), pp.998-1009, 2004.

[2] 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.

[3] 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.

[4] G. Luo, J.F. Naughton, C.J. Ellmann, and M. Watzke, *"Locking Protocols for Materialized Aggregation Join Views,"* IEEE Trans. Knowledge and Data Eng., vol. 17, no.6, pp. 796-807, June 2005.

[5] 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

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

[8] XiangLian, Student Member, IEEE, and Lei Chen, *General Cost Models for Evaluating Dimensionality Reduction in High-Dimensional Spaces*. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22(1):139–144, 2010.

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

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

[11] 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.

[12] C. Cunningham, G. Graefe, and C.A. Galindo-Legeria, *"PIVOT AND UNPIVOT: Optimization and Execution Strategies in an RDBMS,"*Proc: 13th Int'l Conf. Very Large Data Bases (VLDS'04), pp.998-1009, 2004.

[13] G. Graefe, U. Fayyed, 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] XiangLian, Student Member, IEEE, and Lei Chen, *General Cost Models for Evaluating Dimensionality Reduction in High-Dimensional Spaces*. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22(1):139–144, 2010.

[15] Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, Gupta, L.Sheng, and S. *Subramanian.Spreadsheets in RDBMS for OLAP*. In Proc. ACM SIGMOD Conference, pages 52–63, 2003.

[16] The IEEE website. [Online]. Available: http://www.ieee.org/