Extended SQL Aggregation for Database Transformation

Archana A. Chaudhari[#1], Harmeet Kaur Khanuja [#2]

[#]*Department of Computer Engineering, MMCOE, Pune, India*

***Abstract*— To prepare a normalized data set from relational database for analysis requires significant efforts and it is time consuming task. The main reason is that, in general the database grows with many tables and views that must be joined, aggregated and transformed in order to build the required data set. As result, most of the SQL queries are written independently multiple times and in disorganize manner, which create problems in database evolution and software maintenance. To address this issue, we propose simple methods to generate SQL code to return aggregated columns in a horizontal tabular layout, where every row corresponds to an observation, instance or point (possibly varying over time) and every column is associated to a one variable or dimension. 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. By providing these standard normalized data-set as an input to the Decision tree generation algorithm for generating Decision tree, similarly we can generate extended ER model.**

***Keywords*— Data mining, Transformation, Aggregation, Data preparation, Pivoting.**

I. INTRODUCTION

The databases used by enterprises cannot be directly used for data mining process. 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 a normalized data set [7] from relational database for analysis requires significant efforts and it is time consuming task, because database has a collection of normalized tables that must be joined, aggregated and transformed in order to build the required unique data set. Existing SQL aggregations having some limitations to prepare normalized data sets because they return only one column per aggregated group. The summary of business data can be given for data mining purposes instead of giving the whole business data, this is the idea behind preparing normalized data set for data mining. As the most of the data mining algorithms require, input the data set with a horizontal layout, where every row corresponds to an observation, instance or point (possibly varying over time) and every column is associated to a one variable or dimension. Each research discipline having different terminology to describe the data set such as in data mining the common terms are point-dimension, statistics literature uses observation-variable, machine learning research uses instance feature.

The main issue is that relational queries in a data mining project create many temporary tables (static) or views (dynamic), which are not represented as entities in an existing ER model, such collection of transformation tables and disconnected queries complicate database management, software development and software maintenance [1]. This approach is related to an extract-load-transform (ELT) process, in which tables are cleaned and transformed after they are loaded into the database, as compare to traditional Extract-Transform-Load (ETL) tools most data transformation happens outside the DBMS, before loading data [1]. In general, tools that perform data reverse engineering do not give an abstract, well defined representation of database transformations computed with relational queries, nor do they have a data set with variables (in a statistical sense) as the target of such transformations.

A. Motivation

To prepare a suitable data set for data mining 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.

Outer joins are essential for the construction of data sets. However, full outer joins are not useful because, the rows of referencing table must be preserved, whereas right outer joins is equivalent as left outer joins. In this approach left outer join is consider as prominent operator to merge tables to build the desired data set. The most widely commonly known aggregation are sum (), count (), min (), max () etc. returns a column over groups of rows. Unfortunately each aggregation functions and operators in SQL have some limitations to build data sets for data mining purposes. The first limitation is data sets that are stored in a relational database (or a data warehouse) come from Online Transaction Processing (OLTP) systems where database schema are highly normalized. But data mining, statistical or machine learning algorithms generally require data in the form of denormalized with aggregation. 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, which are suitable for data mining algorithm. Second, Standard aggregations are hard to interpret when there are many result rows, especially when grouping attributes have high cardinalities. Third, OLAP tools uses SQL code to generate transpose of results (sometimes called PIVOT). Transposition can be more efficient if there are mechanisms combining aggregation and transposition together. With such limitations in mind, in this approach 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 [2]. Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout.

This paper is organized as follows: Literature Survey is discussed in Section 2, Section 3 introduces Scheme

Description, in this approach three methods to evaluate horizontal aggregations using existing SQL constructs. Section 4 given conclusions and directions for future work.

II. LITERATURE SURVEY

There are many proposals that have extended SQL syntax. Carlos Ordonez et. al proposed framework for programming a clustering algorithm with SQL queries is explored in , which shows horizontal layout of the data set enables easier and simpler SQL queries, their optimization have the purpose of avoiding joins to express cell formulas, but are not optimized to perform partial transposition for each group of result rows [5]. Horizontal aggregations are related to horizontal percentage aggregations, the differences between both approaches are that percentage aggregations require aggregating at two grouping levels, require dividing numbers and need taking care of numerical issues (e.g. dividing by zero), horizontal aggregations are more general, have wider applicability and in fact, they can be used as a primitive extended operator to compute percentages [4]. The paper is proposed by Carlos Ordonez, et. al, in which extend an ER model with new entities to represent database transformations and introduced an algorithm to automate the process, extended ER model has two kinds of entities: source entities and transformation entities, which correspond to normalized tables and temporary tables created with SQL queries, respectively [1].

III. PROPOSED SYSTEM

A. SQL code Generation: Extended SQL Syntax

Twe extend standard SQL aggregate functions with a 'transposing' BY clause followed by a list of columns (i:e. R1, . . . ,Rk), to produce a horizontal set of numbers instead of one number. Our proposed syntax is as follows:

```
SELECT L1,...,Lj, H(A BY R1,...,Rk)
FROM S
GROUP BY L1,...,Lj;
```

We believe the subgroup columns (R1, . . . ,Rk) should be a parameter associated to the aggregation itself. That is why they appear inside the parenthesis as arguments, but alternative syntax definitions are feasible. In the context of our work, H() represents some SQL aggregation (e:g. sum(), count(), min(), max(), avg()). The function H() must have at least one argument represented by A, followed by a list of columns.

B. SQL Code Generation: Query Evaluation

This approach, extending the standard SQL aggregation function and build a new class of aggregation called Horizontal aggregation, which produce tables with a horizontal layout. HA (Horizontal Aggregation) can be implemented by using three different methods:

- SPJ (Select-Project-Join) method relies on standard relational operators.
- CASE method relies on the SQL CASE construct.
- PIVOT method uses a built-in operator in a commercial DBMS that is not widely available.

Figure 1 shows the system control flow of entire system for getting data-sets.

1) *SPJ Method:* The Select-project-Join-Aggregation (SPJ) method 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 S_H [2],this method use the Left Outer Join. The left outer join is performed in between two tables i.e. left table and Right table, common fields of both the tables are returned and uncommon fields of left table is returned. There are two basic sub strategies to compute S_H. The first one directly aggregates from S. The second one computes the equivalent vertical aggregation in a temporary table SV grouping by (L1, . . . ,Lm), (R1, . . . ,Rk), Then horizontal aggregations can be computed from S_V , which is a compressed version of S [2]. We will use first sub strategy to compute S_H. Horizontal aggregation using SPJ method require four input parameters to generate SQL code:-

(i) The input table S
(ii) The list of GROUP BY columns L1, . . . ,Lm
(iii) The column to aggregate (A) and
(iv) The list of transposing columns R1, . . . ,Rk.

In this approach an additional table S0 is introduce that will be Outer joined with projected tables to get a complete result set. Table S0 does not have any nonkey column, but it has (L1, . . .,Lm) as primary key.

```
INSERT INTO S0
SELECT DISTINCT L1, . . . ,Lm
FROM (S\SV);
```

Finally, to get SH we need n+1 tables that will join with n left outer joins. The optimized SPJ method code is as follows:

```
INSERT INTO SH
SELECT S0.L1, S0.L2,....,S0.Lm,
S1.A, S2 .A,...,Sn .A
FROM S0
LEFT OUTER JOIN S1
```

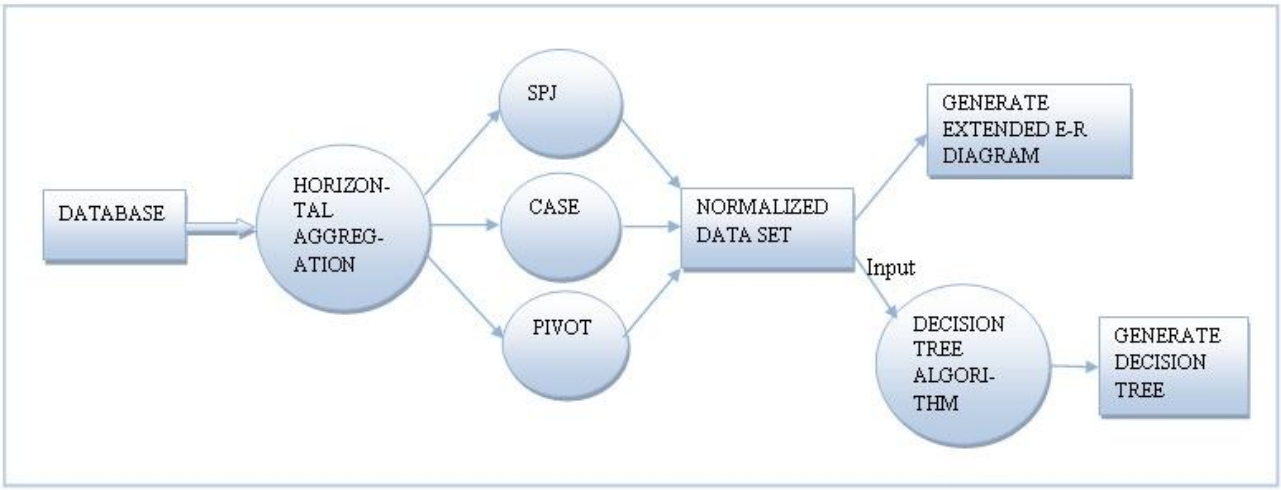

Fig. 1. Data Flow Diagram for Database Transformation

```
ON S0. L1= S1. L1 ...S0. Lm= S1. Lm
LEFT OUTER JOIN S2
ON S0. L1= S2. L1 ...S0. Lm= S2. Lm
. . . .
LEFT OUTER JOIN Sn
ON S0. L1= Sn. L1 ...S0. Lm= Sn. Lm
```

2) *CASE Method:* In this method, 'case' programming construct available in SQL is used. If boolean expressions is satisfied then case statement returns a selected value from a group of values. From a relational database this is equivalent to doing a simple projection/aggregation query where each non-key value is given by a function that returns a number based on some conjunction of conditions [2]. Similar to SPJ method, there are two basic sub strategies to compute S_H. The first one directly aggregates from S. The second one computes the vertical aggregation in a temporary table S_V and then horizontal aggregations are indirectly computed from S_V. The optimized case method code is as follows, where V() is a standard SQL aggregation that has a 'case' statement as a argument.

```
SELECT DISTINCT R1, . . .,Rk
FROM S;

INSERT INTO SH
SELECT L1, . . . ,Lm
,V(CASE WHEN R1 = v11 and . . .Rk = vk1
    THEN A ELSE null END)
...
,V(CASE WHEN R1 = v1d and . . .Rk = vkd
    THEN A ELSE null END)
FROM S
GROUP BY L1, L2, . . . ,Lm;
```

3) *PIVOT Method:* The pivot operator is a built-in operator which transforms row to columns. Since this operator can perform transposition it can help in evaluating horizontal aggregation. The optimized set of queries which reduces the intermediate transposed table is as follows :

```
SELECT DISTINCT R1
FROM S; /*produces v1,,vd*/

SELECT L1, L2,. . . ,Lm
        , v1,v2, . . .,vd
INTO SH
FROM (
SELECT L1,L2,..,Lm, R1,A
FROM S) T
PIVOT(
        V(A) FOR R1 in (v1,v2,,vd)
       ) AS P;
```

C. Extended ER Model Generation

The entity-relationship (ER) model provides diagram notation and methods to design a database, by defining its structure before storing information [1]. In data mining project relational queries create many temporary tables (static) or views (dynamic), which are not represented as entities in an existing ER model. So we extend an ER diagram with entities that represent database transformations used to build data sets.

Algorithm to extend an ER model with transformation entities:

Input: Source tables(S1, S2,. . .) and Queries(q0, q1, q2,. . .)
Output: ER diagram

Step 1: Initialize extended ER model with original ER model.
Step 2: Create a transformation entity j for every intermediate table, numbering it according to query qj in the sequence.
Step 3: For each non-key attribute link it to either a denormalization expression or an aggregation function.

Step 4: Generate the final data-set 'X' which represented as a entity in Extended ER-model.

D. Generate Decision Tree

On the basis of these Datasets Generated from the three methods of Horizontal Aggregation we will generate Decision Tree. This dataset is given as an input to Decision tree algorithm using WEKA to generate Decision tree. On that dataset Entropy and Information gain Operation are performed. Based on that Decision Tree is generated.

Entropy:

$$\text{Entropy}(S) = \sum_{i=1}^{c} (P_i \log_2 P_i)$$

Where, p_i is the probability of class i.

Information Gain:

$$Gain(S, A) = Entropy(S) - \sum_{V \in Values(A)} \frac{|s|}{|s|} Entropy(S)$$

Where, Values (A) is the set of all possible values for attribute A, and s is the subset of S for which the attribute A has value v.

IV. Conclusions

In this approach a new class of aggregate functions in SQL is developed called as horizontal aggregation which is used for preparing data-sets for the data mining projects. Mainly, the existing SQL aggregations return results in one column per aggregated group, but horizontal aggregation returns a set of numbers instead of a single number for each group. Three query evaluation methods are discussed: the first method focuses on the relational operators in SQL, the second method focuses on the SQL case construct and third method focuses on the PIVOT built-in operator in a commercial DBMS. As database grows with many table and view which are not represent as entity in standard ER diagram, so we extending the ER model to generate the Extended ER-diagram. Horizontal aggregation produces tables with fewer rows but with more columns, so the traditional query optimization techniques are inappropriate for the new class of aggregations.

In future work we can develop the most appropriate query optimization technique for the horizontal aggregation to achieve better results.

Acknowledgment

We take this opportunity to thank Prof. Ram Joshi and all the staff members of the Department of Computer Engineering for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this paper.

References

[1] Carlos Ordonez, Sofian Maabout, David Sergio Matusevich, Wellington Cabrera, "Extending ER models to capture database transformations to build data sets for data mining", Data and Knowledge Engineering, vol.89, pp. 38 - 54, January 2014.

[2] Carlos Ordonez and Zhibo Chen,"Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis", IEEE Transaction On Knowledge and Data Engineering, Vol. 24, No. 4, pp. 678-691, April 2012.

[3] Javier Garca-Garcaa, Carlos Ordonez,"Extended aggregations for databases with referential integrity issues", Data and Knowledge Engineering, Vol.69, No.1, pp. 73-95, January 2010.

[4] Carlos Ordonez, "Vertical and Horizontal Percentage Aggregations", Proc. ACM SIGMOD Intl Conf. Management of Data (SIGMOD 04), pp. 866-871, 2004.

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

[7] Carlos Ordonez,"Horizontal Aggregations for Building Tabular Data Sets", Proc. Ninth ACM SIGMOD Workshop Data Mining and Knowledge Discovery (DMKD 04), pp. 35-42, 2004.