International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Horizontal Aggregations in SQL using CASE, PIVOT and SPJ Methods for Data Mining Analysis

[1] A. E. Shewale, [2] P. P. Chouthmal, [3] K. V. Sonkamble, [4] S. N. Deshmukh
[1,2,3,4.] Department of CS and IT, Dr. BAMU, Aurangabad (M.S), India

***Abstract*-Data mining is process of automatically discovering useful information in large data repositories. It is widely used domain for extracting the useful information from large historical data. Data sets used by enterprises can't be directly used for the data mining. That is data sets are to be prepared for real world database to make them suitable for data mining operations. Preparing a data set for analysis is generally the most time consuming task in a data mining project which requires many complex SQL queries with sub queries, aggregating columns, complex joins. Existing SQL aggregation produces the single result per column which result improved by the proposed horizontal aggregation. Horizontal aggregation defines the new class function. It also generates the SQL code and produces the number of output per row. It performs various operations such as CASE, SPJ, and PIVOT. Here, PIVOT operation perform row to column transformation; SPJ is the standard relational algebra operators; CASE method exploring the programming.**

Keywords--SQL Cod; PIVOT; Horizontal aggregation; Preparation of data

I. INTRODUCTION

Aggregation is normally associated with data reduction in relational databases. The aggregate functions available in SQL are MIN, MAX, AVG, SUM and COUNT. These functions returns a single number as output. This aggregation called as a vertical aggregation [1], [3]. The output of vertical aggregations is helpful in calculations and computations. Most of data mining operations require 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 such as regression, classification, PCA and clustering.

A. Motivation

In a relational database, normalized tables a significant effort is required to prepare a summary data set. Every research area uses different terminology to describe a data sets. In data mining common terms are point-dimensions. Statistics literature uses observation variable. Machine learning research uses instance-feature [2]. Here, we are introducing a new class of aggregate functions that can be used to prepare data sets in a horizontal layout for automating SQL query writing and extending SQL capabilities.

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. Horizontal aggregations helps building answer sets in tabular form, which in standard form needed by most data mining algorithms. In horizontal aggregation, a new class of aggregations has similar behaviour to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, standard SQL aggregations that is vertical aggregations which produce tables with a vertical layout. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement.

B. Advantages

Horizontal aggregations provide several unique features and advantages [1]: First, they represent a template to generate SQL code from a data mining tools. Such SQL code automates writing SQL queries, optimizing them and testing them for correctness. Second, since SQL code is automatically generated it is more efficient than SQL code written by an end user. For instance, a person who doesn't know SQL well or someone who is not much familiar with the database schema. Third, the data sets can be created in the less time. Fourth, the data sets can be created entirely inside the DBMS [1].

II. EXISTING SYSTEM

Vertical Aggregations

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 [3]. 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. To overcome the disadvantages of vertical aggregations horizontal aggregations are used.

III. Proposed System

Horizontal Aggregations

Here, a new class of aggregations introduces that have similar behaviour to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, we call standard SQL aggregations vertical aggregations since they produce tables with a vertical layout [1]. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement.

As horizontal aggregations are capable of producing data sets that can be used for real world data mining activities. In this project simple, yet powerful, methods is use to generate SQL code to return aggregated columns in a horizontal tabular

layout, returning a set of numbers instead of one number per row.

A. Methods

Horizontal aggregation is evaluated using three fundamental methods: case, SPJ (Select Project Join) and pivot [1], [3].

1) CASE Method: 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.

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. 1. Example of F, FV and FH

Fig. 1 [1] gives an example showing the input table F, a traditional vertical sum () aggregation stored in FV and a horizontal aggregation stored in FH [1].

Here, the fact table F is used to perform vertical and horizontal aggregation by using the CASE method code is as follows (computed from Fv):

```
INSERT INTO FV
SELECT D1,D2,SUM(A)
FROM F
GROUP BY D1,D2
ORDER BY D1,D2;

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 FV
GROUP BY D1;
SELECT * FROM FH
```

We can compute FH from either F or FV, but we use F to make code more compact.

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) 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 table containing horizontal aggregations. This method performs Select, Project and Join operation on the fact table.

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 F1.D1, F1.A AS D2_X, F2.A AS D2_Y
FROM F1
LEFT OUTER JOIN F2 ON F1.D1=F2.D1;
```

3) PIVOT Method: 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.

The PIVOT method code 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 pvt;
```

B. *Work Flow of Horizontal Aggregation*

1) Previous Work Flow

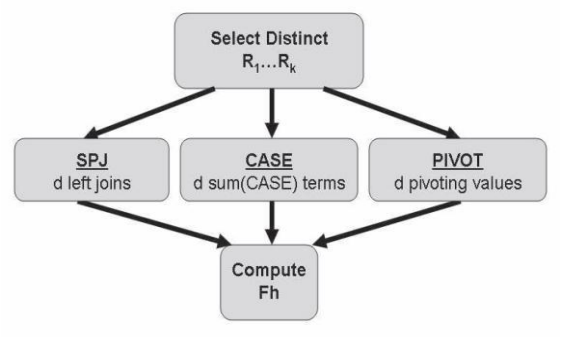

Fig. 2. Main Steps of Methods Based on F *(unoptimized)*

In above Fig. 2 [1], it shows the flow of the horizontal aggregation in which R1,……..,Rk are columns which are used to perform the operations such as SPJ, CASE and PIVOT for d dimensionality. In this diagram, CASE method performs one scan operation. Here it computes aggregation directly from input table. But it gives unoptimized solution.

2) Proposed Work Flow

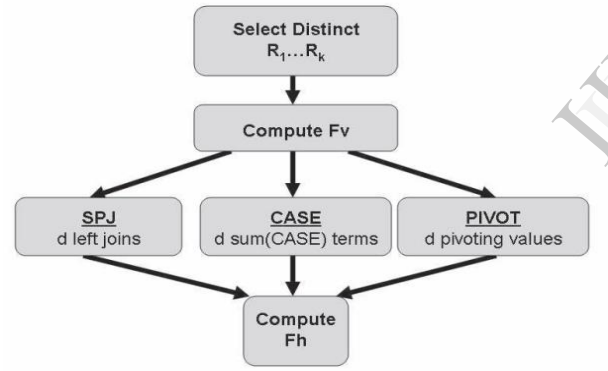

Fig. 3. Main Steps of Methods Based on FV (optimized)

In above Fig. 3 [1], it shows the flow of the horizontal aggregation in which R1,……..,Rk are columns which are used to perform the operations such as SPJ, CASE and PIVOT for d dimensionality. In this diagram, CASE method performs two scan operations.

Here it computes vertical aggregation first and then computes horizontal aggregation. It is necessary to compute vertical aggregation before horizontal aggregation to obtain optimized solution.

IV. EXPERIMENTAL EVALUATION

A. *Data Sets*

We evaluated optimization strategies for aggregation queries with synthetic data sets generated by the TPC-H generator[5]. In general, we evaluated horizontal aggregation queries using the fact table transaction Line as input.

The TPC Benchmark™ H (TPC-H) is a decision support benchmark [5]. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that examine large volumes of data; execute queries with a high degree of complexity.

B. *Query Optimization*

TABLE I QUERY OPTIMIZATION

n	d	CASE		SPJ		PIVOT	
		F	***FV***	***F***	***FV***	***F***	***FV***
100k	2	63	15	130	29	100	15
200k	2	123	30	159	56	150	30

Precompute Vertical Aggregation in Fv. Time in Milliseconds

In above table I, n is the cardinality in rows for the input table and d indicate the no. of dimensions. From table I it is conclude that if we compute the horizontal aggregation after vertical aggregation then it require less time than the horizontal aggregation computed directly by input table. CASE, PIVOT and SPJ method produced same output but they required different time span to compute the query. CASE and PIVOT method require almost same time for both direct and indirect method. But SPJ method require more time than other methods.

V. CONCLUSIONS

In this project a new class of extended aggregate functions introduces which are called as horizontal aggregations which help preparing data sets for data mining and OLAP cube exploration.

In this paper, we use the horizontal aggregation methods CASE, PIVOT and SPJ with direct and indirect method. As we perform CASE method directly from fact table, then it make code more compact but it is time consuming process comparing with indirect CASE evaluation where, horizontal aggregation is computed after vertical aggregation.

Query optimization is most challenging task in the horizontal aggregation. We can try to achieve better query optimization. We can also use the horizontal aggregation for the further extended for Association Rules by applying Apriori Algorithm.

ACKNOWLEDGMENT

The authors would like to thank the University Authorities for providing the infrastructure to carry out the research. This work is supported by University Grants Commission.

REFERENCES

[1] Carlos Ordonez, Zhibo Chen. "Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis," IEEE Transactions on Knowledge and Data Engineering, Digital Object Indentifier 10.1109/TKDE.2011.16, April 2012.

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

[3] V. Pradeep Kumar, Dr. R. V. Krishnaiah. "Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis," ISSN: 2278-0661, ISBN: 2278-8727 Volume 6, Issue 5, PP 36-41,Dec.2012.

[4] C.Ordonez."Vertical and horizontal percentage aggregations." In Proc.ACM SIGMOD Conference, pages 866–871, Oct.2004.

[5] Data sets: http://www.tpc.org/tpch/

[6] G. Bhargava, P. Goel, and B.R. Iyer. "Hypergraph based reordering of outer join queries with complex predicates." In ACM SIGMOD Conference, 1995.

[7] 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.

[8] 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.