Volume 4, Issue 7, July 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Horizontal Aggregation in SQL using CASE, SPJ and PIVOT Method to Prepare Datasets for Data mining

Ankita E. Shewale*
Department of CS and IT,
Dr. BAM university, Aurangabad, India

Dr. S. N. Deshmukh
Department of CS and IT,
Dr. BAM university, Aurangabad, India

Abstract— Data mining is widely used domain for extracting the useful information from large historical data. We can't use the database directly for the data mining which is used by the various enterprises. Preparing a data set for analysis is generally the most time consuming task in a data mining project which requires many complex SQL queries, and complex operations such as joining tables and aggregating columns. 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 automatically 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 which performs two scan method for the proposed horizontal aggregation.

Keywords— SQL Code, Horizontal Aggregation, Preparation of data, PIVOT

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[1]. All these functions returns a single number as output. This is called vertical aggregation. The output of vertical aggregations is helpful in calculation. Most of the 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 like PCA, regression, classification, and clustering.

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. Here, for horizontal aggregation introduces a new class of aggregations that have similar behaviour to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, we call standard SQL aggregations that are 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.

II. LITERATURE SURVEY

Data Mining, also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. It is the process of analyzing useful information that can be used to increase revenue, costs or both. Data mining software is one of the numbers of analytical tools for analyzing data which allows users to analyze data from many different dimensions, angles, categorize it and summarized the relationships.

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed.

To prepare summarized format for data mining algorithm, many methods are introduced by researchers in the past. Carlos Ordonez, Zhibo Chen[1] introduces horizontal aggregations in sql to prepare data sets for data mining analysis. This paper gives the information about three methods of horizontal aggregation. K. Anusha, P. Radhakrishna, and P. Sirisha[2] gives information about horizontal aggregation using spj method and equivalence of Methods. Krupali R. Dhawale and Vani A. Hiremani[14] gives fundamental methods to evaluate horizontal aggregation in sql, where it gives comparison of three horizontal aggregation methods.

SQL has been around since its inception and being used widely for interacting with relational databases both for storing and retrieving data. The SQL provides all kinds of constructs such as projections, selections, aggregations, joins and sub queries. Query optimization and using the result of query further is an essential task in database operations. As part of queries, aggregations are used to get summary of data. Aggregate functions such as SUM, MIN, MAX, COUNT, and AVG are used for obtaining summary of data [1], [3]. These aggregations produce a single value output and can't provide data in horizontal layout which can be used for data mining operations. In other words, the vertical aggregations can't produce data sets for data mining. Association rule mining is one of the problems pertaining to OLAP processing. SQL aggregate functions are extended for the purpose of association rule mining in. The aim of this is to support data mining operations efficiently. The drawback of this is that it is not capable of producing results in tabular format with horizontal layout convenient for data mining operations. In a clustering algorithm is explored which makes use of SQL queries internally.

Traditional query optimizations use tree-based plans for optimization. This is similar to SPJ method. CASE is also used with SQL optimizations. PIVOT in SQL is used for pivoting results. Lot of research has been around on aggregations and optimizations of SQL operations. They also include cross tabulation and explored much in case of cube queries. PIVOT and UNPIVOT are two operators on tabular data that exchange rows and columns, enable data transformation useful in data modeling, data analysis and data presentation [1], [3]. They can quite easily be implemented inside a query processor, much like select, project and join. Such a design provides opportunities for better performance, both during query optimization and query execution. For data mining operations that produce decision trees, vertical aggregations can be used while the horizontal aggregations produce more convenient horizontal layout that is best suited for data mining operations. In SQL Server both pivot and unpivot operations are made available.

A. Vertical Aggregation

This aggregation uses aggregate functions supported by SQL are SUM, MIN, MAX, COUNT and AVG. These functions produce single value output[5]. The result of vertical aggregations is useful in calculations or computations. However, they can't be directly used in data mining operations further. Existing SQL aggregations have limitations to prepare data sets because they return one column per aggregated group.

1) Disadvantages of Vertical aggregation

i) Existing SQL aggregations have limitations to prepare data sets.
ii) To return one column per aggregated group

Group function returns a result based on a group of rows. The functions are generally mathematical functions. To find out the single value results like sum of test scores of students, finding minimum and maximum test score of student these kind of operations can be done by group function.

2) Methods of vertical aggregation: Aggregate functions are a special category with different rules. These functions calculate a return value across all the items in a result set, so they require a FROM clause in the query:

For example

```
select count(product_id) from product_catalog;
select max(height), avg(height) from census_data where age > 20;
```

There are various methods of vertical aggregation:

i) SUM(): An aggregate function that returns the sum of a set of numbers. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, SUM returns NULL.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

```
SELECT student_name, SUM(test_score)From student
GROUP BY student_name
```

ii) MIN(): An aggregate function that returns the minimum value from a set of numbers. Opposite of the MAX function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, MIN returns NULL.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Example:

```
SELECT student_name, MIN(test_score)From student
GROUP BY student_name
```

The example gives the result as a student name that has secure minimum test score.

*iii) MAX():*An aggregate function that returns the maximum value from a set of numbers. Opposite of the MIN function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value.

Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MAX are NULL, MAX returns NULL.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Example:

```
SELECT student_name, MAX(test_score)From student
GROUP BY student_name
```

The example gives the result as a student name that has secure maximum test score.

iv) COUNT(): An aggregate function that returns the number of rows, or the number of non-NULL rows

Example:

```
Select COUNT(*) from student;
```

The result of above syntax measures the number of rows in the student table.

```
Select COUNT(comm) from emp;
```

The result of above syntax count commission amount.

v) AVG: An aggregate function that returns the average value from a set of numbers. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to AVG are NULL, AVG returns NULL.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Example:

```
SELECT student_name, AVG(test_score)From student
GROUP BY student_name
```

B.Horizontal Aggregation

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 [2]. This operation is needed in a number of data mining tasks, such as data summation and unsupervised classification, as well as segmentation of large heterogeneous data sets into the small homogeneous subsets those can be easily managed, separately analyzed and modeled. Then to create datasets for data mining related works, efficient and summary of data will be needed. The main goal of horizontal aggregation is to define a template to generate SQL code combining aggregation and pivoting (transposition). The second goal is to extend the SELECT statement with a "Group By" clause that combines the transposition with aggregation.

Horizontal aggregations provide several unique features and advantages [1]: 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. 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. For instance, a person who does not know SQL well or someone who is not familiar with the database schema. Third, the data sets can be created in less time. Fourth, the data set can be created entirely inside the DBMS.

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

i) *CASE:* For this method we 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 non-key value is given by a function that returns a number based on some conjunction of conditions.

Horizontal aggregation queries can be evaluated by directly aggregating from F that is from fact table and transposing rows at the same time to produce F_H. First, we 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.

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;
```


ii) *SPJ:* 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;
```

iii) 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.

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;
```

2) Previous Process Flow

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_H

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

Fig.1 Example of Horizontal Aggregation from Fact Table

Fig.1 gives an example showing the input table F, and a horizontal aggregation stored in FH[1]. Table F is input table where K shows number of row, D1 is integer values, D2 consists of two values X and Y. Table F is aggregated using horizontal aggregation and this horizontal aggregation is shown in the table FH. In table FH, column A is aggregated and we get the aggregation table which require less space as compare to the input table.

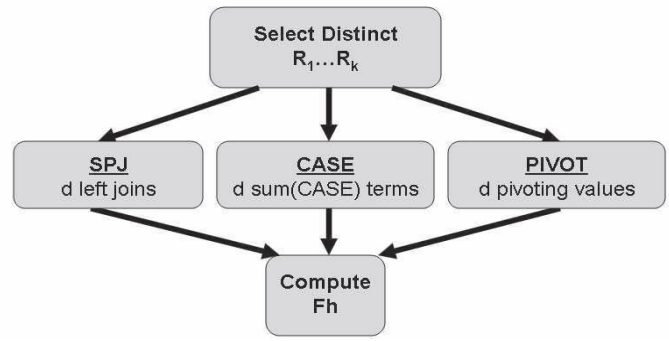

Fig.2 Main Steps of Methods Based on F (unoptimized)

In above Fig 2 [1], previous process flow is shown, the flow of the horizontal aggregation in which $R_1, \ldots\ldots, R_k$ 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. PIVOT method performs operation such as cross tabulation. SPJ method is used to joining two or more table.

III. PROPOSED SYSTEM

A. Proposed Process Flow

Optimizing a workload of horizontal aggregation queries, modifying the query optimizer, Applying Horizontal aggregation to Holistic function are most challenging tasks in horizontal aggregation for further research purpose. For optimizing workload and reducing the time to execute query in this project we first perform vertical aggregation using input table and from vertically aggregated table we compute the horizontal aggregation. Using previous method we can compact the code but to execute the query require more time than proposed system.

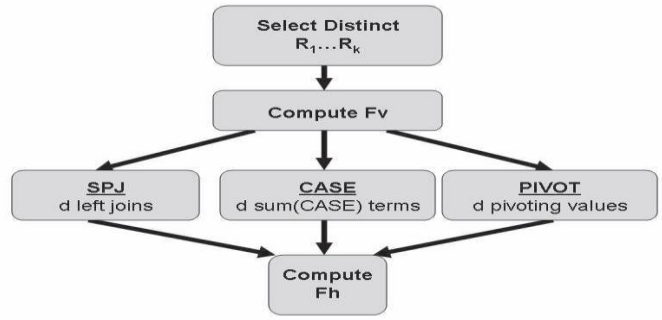

Fig. 3 Main Steps of Methods Based on Fv(optimized)

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 it computes horizontal aggregation. It is necessary to compute vertical aggregation before horizontal aggregation to obtain optimized solution.

In Fig. 4, F is input table, Fv is vertically aggregated table and F_H is horizontally aggregated table. Horizontal aggregation is performing from vertically aggregated table Fv. Table F is input table where K shows number of row, D1 is integer values, D2 consists of two values X and Y. Table F is aggregated using vertical aggregation first and this vertically aggregated table is then converted to horizontally aggregated table and we get the aggregation table which require less space as compare to the input table.

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.4 Example of Horizontal Aggregation from Vertical Aggregated Table

1) CASE: CASE in SELECT statement provides developer the power to manipulates the data at presentation layer without changing data at backend.
Syntax for CASE method:

```
CASE InputValue
WHEN when_value THEN ReturnValue1
WHEN when_value THEN ReturnValue2
        ….
WHEN when_value THEN ReturnValuen
```

Two basic strategies to compute horizontal aggregations: The first strategy is to compute directly from input table. Previously, the horizontal aggregation is computed directly using input table. The second approach is to compute vertical aggregation and save the results into temporary table and then horizontal aggregation is computed by taking input from vertical table. In proposed system we use second approach of CASE method to compute horizontal aggregation.

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

2) SPJ: SELECT is used to obtain a subset of the tuples of a relation that satisfy a select condition. The PROJECT operation is used to select a subset of the attributes of a relation by specifying the names of the required attributes.

The join concept has come due to need of data from different table at a time. To retrieve data from one table is not difficult as getting data from multiple tables either on the basis of common field in those tables or on the basis of particular conditions satisfaction.

3) PIVOT: The pivot or pivot element is the element of a matrix, or an array, which is selected first by an algorithm, to do certain calculations. In the case of matrix algorithms, a pivot entry is usually required to be at least distinct from zero, and often distant from it; in this case finding this element is called pivoting. Pivoting may be followed by an interchange of rows or columns to bring the pivot to a fixed position and allow the algorithm to proceed successfully, and possibly to reduce round-off error.

Pivot transforms a series of rows into a series of fewer rows with additional columns. Data in one source column is used to determine the new column for a row, and another source column is used as the data for that new column. PIVOT operates on a table, like other operations, converting from narrow form to wide form.

PIVOT 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,
[X] as D2_X,
[Y] as D2_Y
FROM
(
SELECT D1, D2, A FROM Fv
) as p
PIVOT
(
SUM(A)
FOR D2 IN ([X], [Y])
)
as pvt;
```

These three methods produces the same result but to compute the result they require different time.

IV. EXPERIMENTAL EVALUATION

A. Setup: Computer Configuration and Data Sets

We used SQL Server v9, running on a DBMS server running at 2.5GHz, core i5 processor, 4GB of RAM and 1 TB hard disk. The SQL code generator was programmed in a C#.net language and connected to the server.

Here, we used large synthetic datasets which is generated by TPC-H generator. We analyzed queries and only one horizontal aggregation, with different grouping and horizontalization columns. Each experiment repeated five times and we report the average time in milliseconds. In general we evaluate horizontal aggregation queries using different tables and TPC-H generator as input for evaluating time require to evaluate aggregation queries.

B. Query Optimization

Table I analyses our first query optimization, applied to three methods. This optimization provides a different gain, depending on the method: for SPJ optimization is best for small n, for PIVOT for large n and for CASE there is a less dramatic improvement all across n.

Table I
Query Optimization: Precompute Vertical Aggregation in Fv. Time in Milliseconds

		SPJ		PIVOT		CASE	
N	D	F	Fv	F	Fv	F	Fv
8k	2	140	1.33	123	0.98	122	1
8k	5	172	1.97	164	1.16	160	1.1
8k	5	198	1.19	165	0.99	162	0.96
16k	5	215	2.06	203	1.39	198	1.32
8k	10	514	2.09	492	1.98	480	1.95
16K	25	1023	32.51	967	25.56	953	25.48

Table II
Comparing Query Evaluation Methods (All with optimization computing Fv). Times in Milliseconds

N	D	SPJ	PIVOT	CASE
8k	2	1.33	0.98	1
8K	5	1.97	1.16	1.1
8K	5	1.19	0.99	0.96
16k	5	2.06	1.39	1.32
8K	10	2.09	1.98	1.95
16 k	25	32.51	25.6	25.48

C. Comparing Evaluation Methods

Table II compares the three query optimization methods. Table II is summarized version of Table I, showing best time for each method. Table III is showing time variability around mean time μ for times reported in table; Here we shows one standard deviation σ and percentage that one σ respect to μ. As can be seen, time exhibit small variability.
The standard deviation is calculated using the following formula:

$$\text{Standard Deviation } (\sigma) = \sqrt{1/n \sum (X_i - \mu)^2}$$

Where,
Σ= Sum of
X_i = Individual score
μ = Mean of all scores
N = Sample size

The percentage of mean for different methods of horizontal aggregation is calculated using following formula:

1) CASE method: (X / (X + Y + Z)) * 10
2) PIVOT Method: (Y / (X + Y + Z)) * 10
3) SPJ Method: (Z / (X + Y + Z)) * 10

Table III
Variability of Mean Time (One Standard Deviation, Percentage of Mean Time). Times in Milliseconds

		CASE		PIVOT		SPJ	
N	d	S.D.	% Of Mean	S.D.	% Of Mean	S.D.	% Of Mean
8k	2	4.21	1.93	4.33	1.89	8.54	6.18
8k	5	5.72	1.75	5.37	1.85	11.09	6.4
8k	5	8.7	2.66	7.77	2.73	16.47	4.61
8k	10	6.74	2.9	5	3.01	11.74	4.09
16k	25	13.68	3.05	13.22	3.06	26.9	3.89

D. Clock Percentage

In this project core i5 processor is used with speed 2.5 GHz. And according to the speed of processor clock percentage cycle is drawn which is shown in below figure. Figure shows that when CASE and PIVOT method executed directly from the input table required less clock percentage than when we execute same methods from vertically aggregated table and for SPJ method clock percentage is vice versa compare to other methods.

Table IV: Clock Percentage for 2D

CASE METHOD	31.67
PIVOT METHOD	31.93
SPJ METHOD	36.34

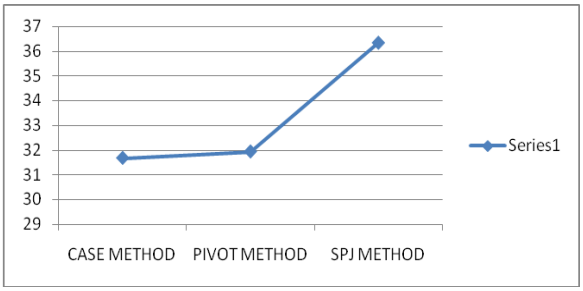

Fig. 5 Clock Percentage for 2D

Table V: Clock Percentage for 5D

CASE METHOD	26
PIVOT METHOD	27.42
SPJ METHOD	46.57

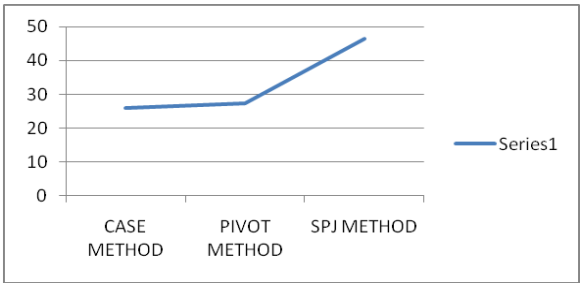

Fig. 6 Clock Percentage for 5D

Table VI: Clock Percentage for 10D

CASE METHOD	32.39
PIVOT METHOD	32.89
SPJ METHOD	34.71

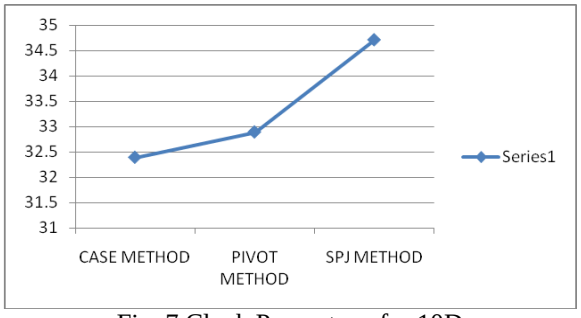

Fig. 7 Clock Percentage for 10D

E. Time Complexity

Here, we plot the time complexity keeping varying one parameter and the remaining parameters fixed. Fig. 9 shows the impact of increasing the size of the fact table (N). In this Figure N grows and the other sizes(n,d) are fixed. According to Fig. 8, The PIVOT and CASE methods show very similar performance. On the other hand, there is big gap between PIVOT/CASE and SPJ. As size of N increases, time requires for evaluating the query is increases. There is big gap between evaluation times for the size of N: 10M and 13M.

Table VII: Impact of Increasing the Size of the Fact Table (N).

N	SPJ	PIVOT	CASE
10M	1.19	0.99	0.96
13M	1.97	1.16	1.1

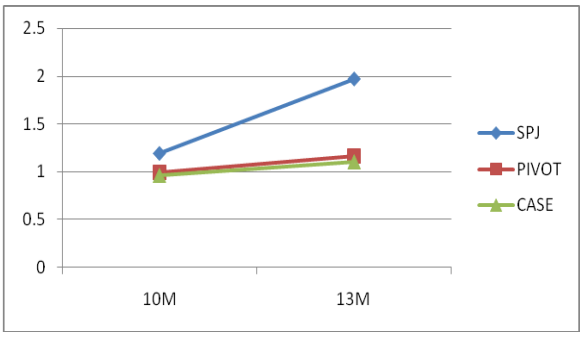

Fig. 8 Time Complexity Varying N

Fig. 9 shows time complexity varying d. Here, PIVOT and CASE methods have almost same performance.

Table VIII: Impact of Varying d

D	SPJ	PIVOT	CASE
5d	1.19	0.99	0.96
10d	2.09	1.98	1.95

Fig. 9 Time Complexity varying d

Fig. 10 shows time complexity varying n (size of horizontally aggregated table). Here, PIVOT and CASE methods have the slightly different performance. And SPJ method requires more time to execute query as compare to CASE and PIVOT method.

Table IX: Impact of Varying Size of Horizontally Aggregated Table (n)

N	SPJ	PIVOT	CASE
8K	1.97	1.16	1.1
16K	2.06	1.39	1.32

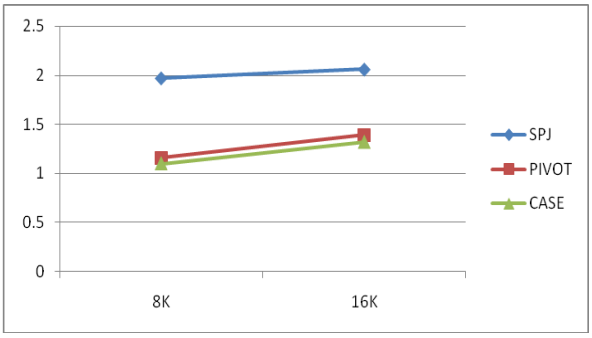

Fig. 10 Time Complexity varying N

V. CONCLUSIONS AND FUTURE WORK

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.

Basically, a horizontal aggregation returns a set of numbers instead of a single number for each group, resembling a multi-dimensional vector. We compare here three query evaluation methods for horizontal aggregation. 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. It is in general the most efficient evaluation method and it has wide applicability since it can be programmed combining GROUP-BY and CASE statements.

As we perform CASE, PIVOT and SPJ methods directly from fact table, then it makes code more compact but it is time consuming process comparing with indirect CASE, PIVOT and SPJ methods evaluation where, horizontal aggregation is computed after vertical aggregation. CASE, PIVOT and SPJ methods produces the same result but they require different time to execute. In project we compare this three methods consisting different execution time. According to the result, CASE and PIVOT methods requires same time to execute horizontal aggregation from vertical aggregation and SPJ method require more time to execute than CASE and PIVOT method.

As graph shown in this project, it is conclude that time require to execute query vary with the changes in the size of fact table(N), size of horizontal aggregated table(n) and dimensions(d).

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 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, 2012.

[2] K. Anusha, P. Radhakrishna, and P. Sirisha, "Horizontal Aggregation using SPJ Method and Equivalence of Methods", March 2012.

[3] V. Pradeep Kumar, Dr. R. V. Krishnaia, ."Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis", Nov. - Dec. 2012.

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

[5] http://jpinfotech.blogspot.in/2011/09/horizontal-aggregations-in-sql-to.html

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

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

[8] 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.

[9] A. Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, A. Gupta, L. Sheng and S. Shubramanian, "Spreadsheets in RDBMS for OLAP", In Proc. ACM SIGMOD Conference, pages 52-63, 2003.

[10] Jincy Annie V. V, J. A. M Rexie, "Evaluating Aggregation Function with Partial Aggregations", International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)Volume 2, Issue 2, February 2013.

[11] A.Lakshman Rao, V.V.Satyanarayana Murty.S, "An Experimental Analysis using PIVOT Method in Data Mining", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 11, November 2012.

[12] Lavina D. Panjwani, Richa K. Makhijani, "Performance Evaluation of Horizontal Aggregation Techniques in Sql", International Journal of Computer Science and Engineering (IJCSE)ISSN 2278-9960Vol. 2, Issue 3, July 2013, 27-34.

[13] Sonali Karle, Swati Avhad, Ashlesha Shelar, Prof. Suvarna Pawar, Kajal Dighe, "Datasets Preparation in SQL using Horizontal Aggregation", International Journal for Research in Applied Science and Engineering Technology (IJRASET), Vol.2 Issue IV, April 2014.

[14] Krupali R. Dhawale 1, Vani A. Hiremani 2,"Fundamental methods to evaluate horizontal aggregation in SQL", International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 10, October 2013.

