IJCSN International Journal of Computer Science and Network, Volume 6, Issue 5, October 2017
ISSN (Online) : 2277-5420
www.IJCSN.org
Impact Factor: 1.5

A Hybrid Approach for Horizontal Aggregation Function Using Clustering

[1] Dr.K.Sathesh Kumar, [2] Dr.S.Ramkumar

[1]Assistant Professor, Department of Computer Science and Information Technology,

[2] Assistant Professor, Department of Computer Applications,
Kalasalingam University, Krishnankoil, Virudhunagar (Dt). INDIA.

Abstract - The research work proposes a very simple and effective summarization based dynamic join operations over high dimensional dataset. These extents the SQL aggregate functions to produce aggregations in horizontal form, returning a set of numbers instead of single aggregation. This work also proposes a Weighted PCA method to handle a high dimensional dynamic dataset with summarization technique. In the proposed technique, there are two common data preparation tasks are enlightened which includes transposition/aggregation and transforming categorical attributes into summarized labels. This executes the basic methods to evaluate horizontal aggregations which are named as CASE, SPJ and PIVOT respectively.

Keywords *aggregation, Weighted PCA method, MCC (Multi class clustering), CASE, SPJ and PIVOT*

1. Introduction

Data aggregation generally works with big data and data marts which will not provide whole information. The issue of data aggregation is occurs when a large amount of data collection on a high security level than individual component of the record. Aggregation is used in dimensional models of the data warehouse to produce dramatic positive effects. Aggregation is a simple summary table that can be grouped by SQL query. The most common use of aggregate is to take dimension and change the granularity of this dimension. Aggregation is referred as pre-calculated summary data. [9] This pre-computing and summarized data are stored in a new aggregated table. When facts are aggregated, it is associated with rolled up dimension. The reason to use aggregation is to increase the performance of the data warehouse in reduction of the number of rows. The aggregation is determined by every possible combination of dimensional granularities. When the request is made by the user the data warehouse should return data from the table with the correct grain [1]. The best way to build the aggregation is to monitor queries and design aggregation to match query patterns. Aggregation data in dimensional model is more complex. To make extra complexity transparent to the user, the functions used to know as aggregate navigation, which is used in query dimensional and fact table. The aggregation navigator is implemented in a range of technology they are as follows:

LAP engines, Materialized views, Relational OLAP server and BI application server or query tools [2].

1.1 Objective of this work

The proposed system aims at developing a new data linkage method aimed at performing horizontal aggregation in high dimensional dataset and one-to-many linkage that can match entities of different types in different tables.
To provide a new data linkage and aggregation method aimed at performing one-to-many linkage.
Using the weighted PCA method, the problem of high dimensional dynamic dataset summarization has been resolved.
Aims at developing a fast and accurate data aggregation and linkage.
The hybrid techniques which is named as Multi Class Clustering (MCC) with PCA.
Aims at reducing the computation overhead.

2. Literature survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support [3].

This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. As horizontal aggregations are capable of producing data sets that can be used for real world data mining activities. This paper presents three horizontal aggregations namely SPJ, PIVOT and CASE. It does mean that we enhanced these operators that are provided by SQL in one way or the other. The SPJ aggregation is developed using construct with standard SQL operations. PIVOT makes use of built in pivoting facility in SQL while the CASE is built based on the [15] SQL CASE construct. We have built a web based prototype application that demonstrates the effectiveness of these constructs [3]. The empirical results revealed that these operations are able to produce data sets with horizontal layout that is suitable for OLAP operations or data mining operations [4] [10].

The motivation behind this work is that developing data sets for data mining operations is very difficult and time consuming. One problem in this area is that the existing SQL aggregations provide a single row output. This output is not suitable for data mining operations. For this reason we have extended the functionalities of CASE, SPJ and PIVOT operators in such a way that they produce data sets with horizontal layouts [4] [5].

In a relational database environment building a suitable data set for data mining purposes is a time-consuming task. This task generally requires writing long SQL statements or customizing SQL if it is automatically generated by some tool. There are two main ingredients in such SQL code: joins and aggregations. We concentrate on the second one. The most widely-known aggregation is the sum of a column over groups of rows. Some other aggregations return the average, maximum, minimum or row count over groups of rows. There also exist non standard extensions to compute statistical functions like linear regression, quintiles and variance. There is even a family of OLAP-oriented functions that use windows and row partitioning. Unfortunately, all these aggregations present limitations to build data sets for data mining purposes.

The main reason is that, in general, data that are stored in a relational database (or a data warehouse to be more specific) come from On-Line Transaction Processing (OLTP) systems where database schemas are highly normalized. But data mining, statistical or machine learning algorithms generally require data in a summarized form that needs to be aggregated from normalized tables. Normalization is a well known technique used to avoid anomalies and reduce redundancy when updating a database [6].

When a database schema is normalized, database changes (insert or updates) tend to be localized in a single table (or a few tables). This helps making changes one row at a time faster and enforcing correctness constraints, but it introduces the later need to gather (join) and summarize (aggregate) information (columns) scattered in several tables when the user queries the database. Based on current available functions and clauses in SQL there is a significant effort to compute aggregations when they are desired in a tabular (horizontal) form, suitable to be used by a data mining algorithm. Such effort is due to the amount and complexity of SQL code that needs to be written and tested. To be more specific, data mining algorithms generally require the input data set to be in a tabular form having each point/observation/instance as a row and each dimension/variable/feature as a column. There are further practical reasons supporting the need to get aggregation results in a tabular (horizontal) form. Standard aggregations are hard to interpret when there are many result rows, especially when grouping attributes have high cardinalities. To perform analysis of exported tables into spreadsheets it may be more convenient to have aggregations on the same group in one row (e.g. to produce graphs or to compare subsets of the result set). Many OLAP tools generate code to transpose results (sometimes called pivot). This task may be more efficient if the SQL language provides features to aggregate and transpose combined together. With such limitations in mind, the proposed new class of aggregate functions that aggregate numeric expressions and transpose results to produce a data set in tabular form. Functions in this class are called horizontal aggregations.The above terms and related works already discussed in previous research work [7][8].

3. Methodology

The methodology defines the steps and procedure of the proposed methodology. This chapter clearly defines the weighted PCA implementation with the temporal aggregation. The system additionally implements the Hybrid technique which is named as Multi class clustering which is blended with the CASE, SPJ and PIVOT tools.

3.1 Salient features of the proposed system:

The proposed Multi class aggregation provides several exclusive features and advantages. 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.

These types of implementations reduce 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. And the system supports for the dynamic dataset.
Third, the data set can be created entirely inside the DBMS. In modern database environments, it is common to export de-normalized data sets to be further cleaned and transformed outside a DBMS in external tools (e.g., statistical packages). Unfortunately, exporting large tables outside a DBMS is slow, creates inconsistent copies of the same data and compromises database security. Therefore, we provide a more efficient, better integrated and more secure solution compared to external data mining tools. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement.

Aggregation Process

This introduces a new multi class aggregation that has similar deeds to SQL standard aggregations and produces tables with a horizontal and vertical layout. The SQL aggregation functions just need a small syntax extension to aggregate functions described 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. The chapter provides the automatic generation of SQL code from the front end using data mining technique.

The main process is to define a template to generate SQL code combining aggregation and transposition (pivoting). A second goal is to extend the SELECT statement with a clause that combines transposition with aggregation.

Weighted PCA Aggregation techniques and proposed Multi Class Clustering (MCC) techniques already elaborated in our previous research work on Horizontal Aggregation Function Using Multi Class Clustering (MCC) and Weighted (PCA) [12,7].

3.2 Temporal Aggregation

Under the aggregation process, the temporal aggregation has been applied. The temporal aggregation is the process of aggregating the data in the timely manner. This splits the data according to the date of updatation. This helps to avoid database scanning every time.

In temporal databases, temporal grouping is a process where the time-line is partitioned over time and tuple's are grouped over these partitions. Then, aggregate values are computed over these groups. This temporal aggregation can be processed in a sequential or parallel fashion. The parallel processing technology becomes even more attractive as the size of data-intensive applications grows as evidenced in OLAP and data warehousing environments.

In this process this proposes a variety of temporal aggregation algorithms along with the WPCA that overcome major drawbacks of previous work.

3.3 Overall Proposed Architecture

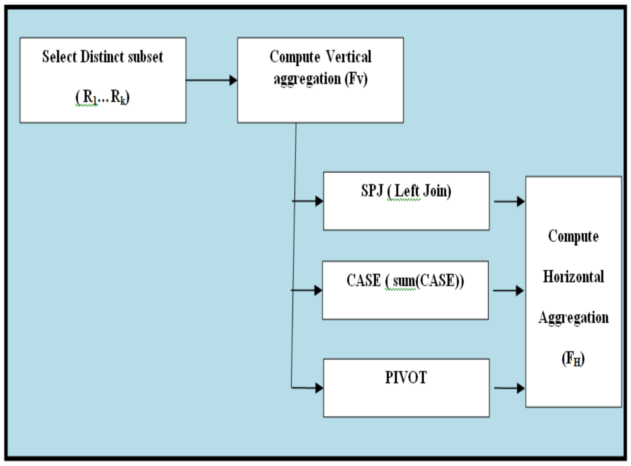


Fig 1.1 Process of aggregation using CASE, PIVOT and SPJ

4. Method description

The implementation of the proposed idea intends the basic three methods to evaluate horizontal aggregations. The first method relies only on relational operations. That is, only doing select, project, join, and aggregation queries; that is known as SPJ method. The second method relies on the SQL "case" constructs; this is also known as CASE method. Each table has an index on its primary key for efficient join processing. This does not consider additional indexing mechanisms to accelerate query evaluation. The third method uses the built-in PIVOT operator, which transforms rows to columns which is also known as transposing.

4.1 SPJ Method

SPJ is based on standard relational algebra operators, which is expanded as (Select Project Join). 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 SPJ method is more reliable based on theoretical perspective because it is based on relational operators only. The basic scheme is to create one table with a vertical aggregation for each result column, and then join all those tables to produce the modified table. This aggregate function performs Select-Project-Join-Aggregation queries (selection, projection, joins aggregation). [13][14]

1. Introduce an additional table F0
2. Perform the following and store the result Rs in F0.
    a. Perform outer join with projected tables.
3. Computation of horizontal aggregation (FH)
    a. A simple table F with a primary key P. aggregate from F.
    b. Computes the equivalent vertical aggregation in a temporary table FV grouping by L1,………,Lj, R1,………,Rk.
4. Perform horizontal aggregations computed from FV, which is a compressed version of F.

This initiates the indirect aggregation based on the intermediate table FV , that will be used for both the SPJ and the CASE method. Let FV be a table containing the vertical aggregation, based on L1; . . . ; Lj;R1; . . .;Rk. Let V() represent the corresponding vertical aggregation for Horizontal aggregation functions. The statement to compute FV gets a cube:

```
INSERT INTO FV
SELECT L1; . . . ; Lj;R1; . . .;Rk, V(A)
FROM F
GROUP BY L1; . . . ; Lj;R1; . . .;Rk;
```

Table F0 defines the number of result rows, and builds the primary key. F0 is populated so that it contains every existing combination of L1,……,Lj. Table F0 has {L1,…….., Lj} as primary key and it does not have any non-key column.

```
INSERT INTO F0
SELECT DISTINCT L1,……, Lj
FROM {F|FV};
```

Step 1: create a table F0

Step 2: select distinct grouping columns (L1….Lj) from first table or vertically aggregated table.

Step 3: Generate Boolean expressions for where section

The step 2 represents to get all distinct combinations of sub-grouping columns R1,……..,Rk, to create the name of dimension columns, to get d, the number of dimensions, and the step 3 helps to generate the Boolean expressions for WHERE clauses. Each WHERE clause consists of a combination of k equalities based on R1…Rk.

```
SELECT DISTINCT R1…Rk
FROM {F|FV};
```

Tables F1 . . . ;Fd contain individual aggregations for each combination of R1 . . . Rk. The primary key of table FI is {L1 . . . Lj}.

```
INSERT INTO FI
SELECT L1,……,Lj; V (A)
FROM {F|FV}
WHERE R1 = v_1I AND .. AND Rk = v_kI
GROUP BY L1,……….,Lj;
```

Step 1: create a table FI

Step 2: select distinct grouping columns (L1….Lj) from first table or vertically aggregated table and vertically aggregated data.

Step 3: Generate where condition to evaluate every subset R.

Step 4: evaluate this result by applying a group function.

Finally, to get horizontally aggregated table (FH) this needs d left outer joins with the d + 1 tables so that all individual aggregations are properly assembled as a set of d dimensions for each group.

The following is the syntax for outer join.

SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;

Outer joins set result columns to null for missing combinations for the given group. In general, nulls should be the default value for groups with missing combinations.

```
INSERT INTO FH
SELECT
F0:L1; F0:L2; . . . ; F0:Lj,
F1:A; F2:A; . . . ; Fd:A
```

```
FROM F0
LEFT OUTER JOIN F1
ON F0:L1= F1:L1 and . . . and F0:Lj = F1:Lj
LEFT OUTER JOIN F2
ON F0:L1= F2:L1 and . . . and F0:Lj = F2:Lj
. . .
LEFT OUTER JOIN Fd
ON F0:L1 = Fd:L1 and . . . and F0:Lj = Fd:Lj;
```

Another claim is that it is not possible to correctly compute horizontal aggregations without using outer joins.

4.2 CASE Method

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.

In the implementation process this use the "CASE" programming to construct available in SQL. This proposes two basic sub-strategies to compute FH.

- In a similar manner to SPJ, the first one directly aggregates from F and
- The second one computes the vertical aggregation in a temporary table FV and then horizontal aggregations are indirectly computed from FV. It's now presents the direct aggregation method.

Horizontal aggregation queries can be evaluated by directly aggregating from F and pivoting rows at the same time to produce the horizontal aggregated table (FH).

Step 1: Get the unique combinations of R1; . . .; Rk that define the matching Boolean expression for result columns.

Step 2: perform the following procedure Vertical aggregation V (statement)…Return result (G).

Step 3: if (rows! = qualifying rows (Qr))…Return result as Null.

The SQL code to compute horizontal aggregations directly from F is as follows: observe V () is a standard (vertical) SQL aggregation that has a "case" statement as argument. Horizontal aggregations need to set the result to null when there are no qualifying rows for the specific horizontal group to be consistent with the SPJ method and also with the extended relational model.

Simple CASE expression:

```
CASE input_expression
     WHEN when_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
```

As can be seen, the code is similar to the code presented before in SPJ, the main difference being that this has a call to sum ( ) in each term, which preserves whatever values were previously computed by the vertical aggregation. It has the disadvantage of using two tables instead of one as required by the direct computation from F. For very large tables F computing FV first, may be more efficient than computing directly from F.

4.3 PIVOT Method

The pivot method is an integral method which transforms rows into columns. It internally needs to determine how many columns are needed to store the pivoted table and it can be combined with the GROUP BY clause. Since this operator can perform transposition it can help in evaluating horizontal aggregation. Since this method can perform pivot or also known as transposition it can help assessing horizontal aggregations. The basic syntax to exploit the PIVOT operator to compute a horizontal aggregation assuming one BY column for the right key columns (i.e., k= 1) is as follows:

```
SELECT <non-pivoted column>,
   [first pivoted column] AS <column name>,
   ...
   [last pivoted column] AS <column name>
  FROM
   (<SELECT query that produces the data>) AS <alias for the source query>
PIVOT
( <aggregation function>(<column being aggregated>)
FOR
[<column that contains the values that will become column headers>]
   IN ( [first pivoted column], [second pivoted column],
   ... [Last pivoted column])
) AS <alias for the pivot table>
<Optional ORDER BY clause>;
```

Notice that in the optimized query the nested query trims F from columns that are not later needed. That is, the nested query projects only those columns that will participate in FH.

5. Performance analysis

To evaluate the performance of the proposed schemes, execution time (or “delay”) is the main measurement of performance evaluation. Without loss of generality, this defines processing delay and aggregation delay for the proposed system. Processing delay indicates the execution time for aggregation to produce the summarized result over a set of data. Aggregation delay is also evaluated by measuring time spent on processing time on aggregating horizontal data and as well as vertical aggregation schemes. At last PIVOT delay, CASE operation delay, is not considered since the scheme is considerably combined in the MCC. Therefore, this delay is negligible and can be ignored. Another criterion is accuracy evaluation.

This section analyzes the first query optimization which is applied to three methods. The intension of this implementation is to assess the acceleration obtained by pre-computing a cube and storing it on FV and updating with the summarized data. This shows the MCC optimization uniformly accelerates all methods. This optimization provides a different gain, depending on the method: for SPJ the optimization is best for small number of datasets, for PIVOT the optimization is best for large n and for CASE there is rather a less dramatic improvement all across the number of records. But the MCC is more optimal for large and dynamic data sets. It is noteworthy PIVOT and CASE is accelerated by the optimization, despite the fact it is handled by the query optimizer.

Since this optimization produces major speeding up for the three methods. The experiments show that pre-computing FV takes the same time within each method. but this may vary if the data set id dynamic Therefore, comparisons are fair. This now evaluated an optimization specific to the PIVOT operator. This PIVOT optimization is well known, as we learned from SQL Server DBMS users groups. The followings are the comparison between the existing and the proposed system.

The table 1.1 shows the performance comparison of the proposed method with other existing approaches based on the three different metrics processing delay, aggregation delay, and overall efficiency.

Table 1.1 performance comparison of the proposed method with other existing approaches

METHOS	PROCESSIG DELAY	AGGREGATION DELAY	EFFICIEY
JOINS	3702.09	73.71	476
BAYESIA CLASSIFIR	2984.06	70.68	396
MCC	2108.08	49.69	256

This performance is based on three methods Join function, Bayesia classifier and Multi Class Clustering (MCC). From these three methods MCC gives better results than other two methods.

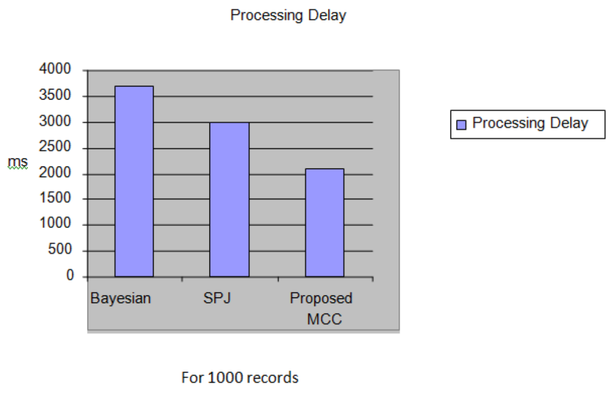


Fig 1.2 Performance comparison of proposed MCC using Weighted PCA and CASE tool with existing approaches based on Processing Delay

From the chart it shows the performance measure based on the processing delay and the proposed approach MCC using WPCA took less time while comparing the other methods and the worst time complexity is Bayesian Classifier.

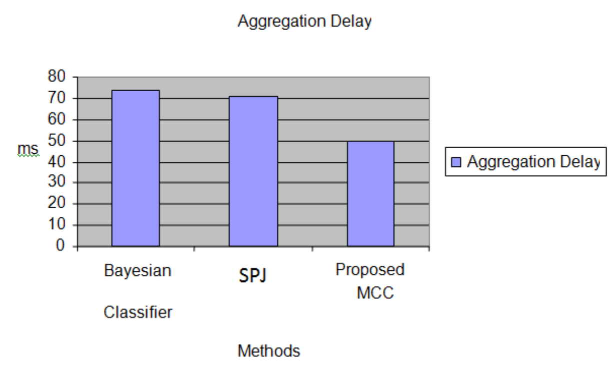
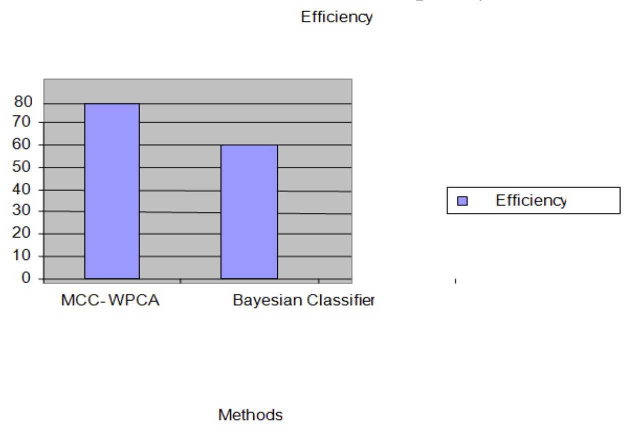


Fig 1.3 Performance comparison of proposed MCC using Weighted PCA and CASE tool with existing approaches based on Aggregation Delay

From the chart it shows the performance measure based on the aggregation delay and our proposed approach MCC using WPCA took less time while comparing the other methods and the worst time complexity is SPJ



1.4 Performance comparison of proposed MCC using WPCA with existing approaches based on Efficiency

From the chart it shows the performance measure based on the Aggregation Energy and our proposed approach MCC using WPCA took more efficiency while comparing the other methods and the worst efficiency is Bayesian classifier.

5. Conclusion and future work

The thesis presented and introduced a new multi class of aggregate functions which is called horizontal aggregations with the innovation of MCC. Horizontal aggregations are useful to build data sets in tabular form when it is huge. A horizontal aggregation returns a set of numbers instead of a single number for each group. This thesis proposed a simple extension to SQL standard aggregate functions to compute horizontal aggregations that only requires specifying sub grouping columns in a dynamic environment. This described how to evaluate horizontal aggregations with multi class clustering values with standard SQL using two best basic strategies. The first one (SPJ) relies on relational operators. The second one (CASE) relies on the SQL case construct. The SPJ strategy is interesting from a theoretical point of view because it is based on select, project, natural join and outer join queries. The CASE strategy is important from a practical standpoint given its efficiency. So the system utilizes the CASE strategy in order to aggregate the data with the help of SQL. The system additionally performs the Weighted PCA method in order to reduce the dimensionality in aggregation. This WPCA slightly reduce the time delay when the aggregations take place. The WPCA maintains the summarized results for further aggregation. The experiments and evaluation shows the proposed Multi class aggregation scheme yields best result in the portion of dynamic environment,

Acknowledgments

We thank the Karpagam University for the motivation and encouragement for giving the opportunity to do this research work as successful one.

References

[1] Chaudhari, A. A., & Khanuja, H. K. (2014). Extended SQL Aggregation for Database Transformation. International Journal of Computer Trends and Technology (IJCTT), 18(6), 272-275.
[2] George, B., & Balaram, A. Dataset Preparation and Indexing for Data Mining Analysis Using Horizontal Aggregations. International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 5, May 2014
[3] Tassa, T. (2014). Secure mining of association rules in horizontally distributed databases. IEEE Transactions on Knowledge and Data Engineering, 26(4), 970-983.
[4] Ordonez, C. (2004, June). Horizontal aggregations for building tabular data sets. In Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (pp. 35-42). ACM.
[5] Saravanan, M., & Mythili, P. A Novel Approach of Horizontal Aggregation in SQL for Data Mining. International Journal of Engineering Trends and Technology (IJETT) – Volume 9 Number 1 - Mar 2014, pp 45-47
[6] Nyaykhor, R. S., & Deotale, N. T. (2015). Horizontal Aggregations In SQL To Generate Data Sets For Data Mining Analysis In An Optimized Manner. *International Journal of Computer Science and Network Security*

(IJCSNS), *15*(3), 24.

[7] K.Sathesh Kumar, P.Sabiya and S.Deepika (2017), Horizontal Aggregation Function Using Multi Class Clustering (MCC) and Weighted (PCA), International Journal of Advanced Research in Computer and Communication Engineering ICITCSA 2017,6(1),pp 199-204.

[8] Ordonez, C., & Chen, Z. (2012). Horizontal aggregations in SQL to prepare data sets for data mining analysis. *IEEE transactions on knowledge and data engineering*, *24*(4), 678-691.

[9] Kumar, V. P., & Krishnaiah, R. V. (2012). Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis. *IOSR Journal of Computer Engineering (IOSRJCE)*, 2278-0661.

[10] Phalak, M. P., & Sharma, R. Optimization of Horizontal Aggregation in SQL by using C4. 5 Algorithm and K-Means Clustering.

[11] PATIL, A. A., & Thakore, D. M. (2014). Optimizing the Query for Preparing the Dataset for Horizontal Aggregation Using Case and Pivot Method. *International Journal of Research in Advent Technology*, *2*(12), 60-65.

[12] Dr.K.Sathesh Kumar and Dr.S.Ramkumar, 2016. A Personalized Scheme For Incomplete And Duplicate Information Handling In Relational Databases", International Journal Of Engineering Sciences & Research Technology, 5(9), 360-369

[13] Sawale, G. J., & Gupta, S. R.(2014) Horizontal Aggregations Based Data Sets for Data Mining Analysis: A Review.

[14] Nyaykhor, R. S., & Deotale, N. T. (2015). Horizontal Aggregations In SQL To Generate Data Sets For Data Mining Analysis In An Optimized Manner. *International Journal of Computer Science and Network Security (IJCSNS)*, *15*(3), 24.

[15] Phalak, M. P., & Sharma, R.(2014) Optimization of Horizontal Aggregation in SQL by using C4. 5 Algorithm and K-Means Clustering.

Author's Biography

Dr. K. Sathesh Kumar completed M.C.A., Ph.D. He is presently working as an Assistant Professor in the Department of Computer Science & Information Technology, Kalasalingam University, Krishnankoil, India He has five years of experience in teaching and research level and also he published many research Papers in both International and National Journals. His research areas include Data Mining, Image Processing, Computer Networks, Cloud Computing, Software Engineering and Neural Network.

Dr.S.Ramkumar is currently working as an Assistant Professor at Kalasalingam University, Krishnan Koil. He received his MCA Degree in Karunya University and M.Phil. Degree in Karpagam University. He has five years of Excellency in teaching and worked as Assistant professor in PG Department of Computer Science at Subramanya College of Arts and Science, Palani, and V.S.B Engineering College, Karur in Tamilnadu. He obtained his Doctorate degree in Computer science in Karpagam University, Coimbatore, Tamilnadu. His areas of interest include Data Structures, Operating System, Java, Web Programming, System Software, Object Oriented Analysis and Design, Software Engineering and Digital Signal Processing. He has published several papers in referred journals and conferences. His field of interest is Bio Signal Processing, Artificial Intelligence, Huma