ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.2 , No.6, Pages : 32-37 (2013)

Special Issue of ICETEM 2013 - Held on 29-30 November, 2013 in Sree Visvesvaraya Institute of Technology and Science, Mahabubnagar – 204, AP, India

CREATING MINIMIZED DATA SETS BY USING HORIZONTAL AGGREGATIONS IN SQL FOR DATA MINING ANALYSIS

Subbarao Jasti[#1], Dr.D.Vasumathi[*2]

[1]*Student & Department of CS & JNTU, AP, India*

[2]*Professor & Department of CSE & JNTU, AP, India*

[1]subbu.jasti@gmail.com

[2] vasukumar_devara@yahoo.co.in

Abstract: Mining historical data can help in discovering actionable knowledge. However, mining activities cannot be done directly on the regular databases. In order to perform data mining, it is required to prepare datasets that will be useful for mining process. Preparing datasets manually for data mining is a challenging task as it needs aggregations, complex SQL queries. The problem with existing aggregate functions of SQL such as SUM, MAX, MIN, COUNT and AVG return a single value as output. Such scalar value cannot be used for preparing datasets. Therefore a new approach is required to generate datasets required for data mining automatically. In this paper we present various techniques to generate datasets for data mining through horizontal aggregations. The proposed aggregations in this paper include PIVOT, SPJ, and CASE. We also built a prototype application to demonstrate the proof of concept. The experimental results revealed that the proposed aggregations are useful in generating datasets for data mining.

Key words : Data mining, data sets, SQL, horizontal aggregations

INTRODUCTION

RDBMS is the stable database model which is widely used in the real world. It is used to store data permanently. Such data is accessed via applications known as front ends. The front end applications make use of SQL to interact with RDBMS for storing and retrieving data. Databases are prepared based on the requirements of an organization. The data is stored in relations and they are normalized to make them more compact and consistent. SQL queries can be used to aggregate data present in tables. The common aggregate functions supported by SQL include COUNT, MAX, MIN, AVG and SUM. All these functions can produce output as a scalar value. They are also known as vertical aggregations. The results of these aggregation functions can be used in summary analysis. However, they are not useful for data mining operations. It does mean that they cannot form a dataset for data mining. Datasets can be prepared for data mining by horizontally aggregating the data present in relational tables [1]. Statistical algorithms [2], [3] can make use of such summary of data. Many existing data mining techniques like classification, PCA, regression, and clustering expect data in the form of tables with tuples and domains [2], [4]. Therefore it is important to use horizontal aggregations in order to generate datasets. In this paper we proposed such aggregations like PIVOT, CASE, and SPJ. These operations are the routines that may use SQL commands internally to generate horizontal aggregations that result in data sets which can be used directly for data mining. Built in facility of pivoting of SQL is used by PIVOT. In the same fashion SPJ and CASE are built with underlying SQL commands to generate data sets for real world data mining.

This work is motivated by the fact that generating datasets can help in data mining directly. This is because the database which is regularly used store data related to OLTP (Online Transaction Processing) which is not suitable for data mining. For this reason, it is better to take data from such databases and generate data sets meant for data mining.

The horizontal aggregations that we have proposed are very useful for generating data sets for data mining operations. Various constructs like SPJ, PIVOT and CASE are prepared in order to generate data sets required. The underlying SQL commands used by these aggregations help in achieving the goal of preparing datasets for data mining. The horizontal aggregation operations we proposed can be used by users.

They in turn generate required SQL code automatically to achieve the goal.

We also built a prototype application which demonstrates the proof of concept. The application is web based and allows the end users to generate datasets using different horizontal aggregations proposed by us. The experimental results revealed that the proposed application is useful in generating real world data sets for data mining purposes. The rest of the paper is organized as follows. Section II reviews literature. Section III provides details about proposed horizontal aggregations.

PRIOR WORK

SQL is widely used in real world applications as it supports interaction with various relational databases. SQL has simple and effective commands to perform operations such as DML, DDL, TCL and DCL. They also support sub queries, joins and aggregations. Optimization of queries is possible by using certain performance tuning activities such as indexing. As part of SQL queries, aggregate functions like MIN, MAX, COUNT, AVG and SUM can be used to get summary of data [5]. The aggregate functions provided by SQL generate single row output. They cannot provide data in horizontal layout which is essential for data mining applications. One of the data mining techniques is association rule mining which is widely used in OLAP processing [6]. In [7] an extension to SQL aggregate functions is made for efficient data mining solutions. The result of such aggregate functions can provide data in horizontal layout which is useful for data mining. Clustering algorithm [5] is one of the data mining algorithms that make use of SQL internally in order to perform clustering. Horizontal layout is used for data which is further used to perform data mining operations through custom SQL extensions [8]. SQL extensions provided by them have optimizations for joins but not for resultant groups. For this reason joins can be avoided using PIVOT and CASE constructs. For new class of aggregations in [9] relational algebra is used. Such aggregations are known as horizontal aggregations. In this paper also we focus on the horizontal aggregations such as CASE, PIVOT and SPJ. Optimizing joins is also presented in [10] but that is not useful for large queries. Query optimizations using tree-based plans are also explored in [11]. There was lot of research on aggregations and related optimizations that include cross tabulation [12]. Exploration of un-pivoting relational tables is done in [13] where decision trees are computed using input rows. The results contain multiple attribute – value pairs for horizontal aggregations. Transforming data from one format to another format can be done using SQL operations [14]. Unpivot operator is similar to TRANSPOSE operator that produce multiple rows. Number of operations is reduced in TRANSPOSE when compared to PIVOT. There is inverse relationship between these two operators. Vertical aggregations and decision trees are used for horizontal aggregations that produce layouts that can be used for data mining operations. SQL Server [15] supports both unpivot and pivot operations are available. In [16] and [17] aggregations are explored but they have some limitations. The results of them can’t be directly used to produce datasets for real time data mining operations.

From built in aggregations of SQL language, the horizontal aggregations are different. In this paper we implement horizontal aggregations such as CASE, SPJ, and PIVOT which are extensions to the built in operators of SQL. For example CASE is a construct which is done programmatically that make use of SQL internally to perform the horizontal aggregations.

This section provides details about horizontal aggregations with suitable illustrations. The aggregations considered include SPJ, PIVOT and CASE.

III.EXECUTION STRATEGIES IN HORIZONTAL AGGREGATION

This section provides details about horizontal aggregations with suitable illustrations. The aggregations propose a new class of functions that aggregate numeric expressions and the result are transposed to produce data sets with a horizontal aggregation. The operation is needed in a number of data mining tasks, such as unsupervised grouping and data aggregation, as well as reduction of large heterogeneous data sets into smaller homogeneous subsets that can be easily managed, separately modeled and analyzed. To create datasets for data mining related works, efficient and aggregation of data are needed. For that this

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.2 , No.6, Pages : 32-37 (2013)
Special Issue of ICETEM 2013 - Held on 29-30 November, 2013 in Sree Visvesvaraya Institute of Technology and Science, Mahabubnagar – 204, AP, India

proposed system collect particular needed attributes from the different fact tables and displayed columns in order to create data in horizontal aggregation. Main goal is to define a template to generate SQL code combining aggregation and pivoting(transposition). A second goal is to extend the SELECT statement with a clause that combines transposition with aggregation. Speculate the following GROUP BY query in standard SQL that takes a subset C1, . . ,Cm fromB1, . ., Bp:

SELECT C1, .., Cm, sum (S) FROM E1,E2 GROUP BY C1,Cm;

In a horizontal aggregation there are 4 input parameters to generate SQL code:

1) The input table E1,E2……,En
2) The list of GROUP BY columns C1, . . ., Cj ,
3) The column to aggregate (S),
4) The list of transposing columns A1, . . .,Ak.

This aggregation query will produce a wide table with m+1 columns (automatically determined), with one group for each different combination of values C1, ... ,Cm and one aggregated value per group (i.e., sum(S)). In order to evaluate this query the query optimizer takes three input parameters. First parameter is the input table E1. Second parameter is the list of grouping columns C1, . . . , Cm. And the final parameter is the column to aggregate (S).

Example

In the Figure.1 there is a common field K in E1 and E2.In E2, B2 consist of only two distinct values P and Q and is used to transpose the table. The grouping operation is used in this is sum (). The values within B1 are repeated, 1 appears 3 times, for row 3, 4 and, and for row 3 & 4 value of B2 is P & Q. So B2P and B2Q is newly generated columns in E_H.

Table E_1

K	B1	B2
1	3	P
2	2	Q
3	1	Q
4	1	Q
5	2	P
6	1	P
7	3	P
8	2	P

Table E_2

K	S
1	9
2	6
3	10
4	0
5	1
6	NULL
7	8
8	7

Table E_H

B1	B2P	B2Q
1	NULL	10
2	8	6
3	17	NULL

Figure 1. An example of Horizontal aggregation

Commonly using Query Evaluation methods in Horizontal aggregation functions [18] are SPJ, PIVOT and CASE methods.

Steps Used in All Methods

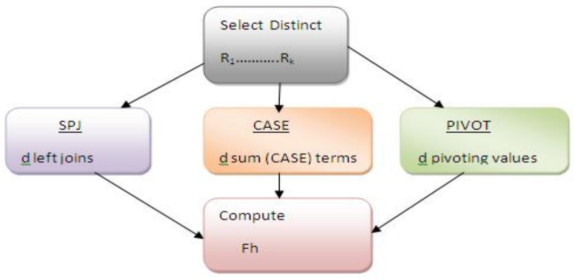

Figure. 2 shows steps on all methods based on input table

All aggregation methods such as PIVOT, CASE, and SPJ proposed in this paper follow the steps as described in fig. 2. For each method the first step is issuing a SELECT query. Then other operator is applied as required. Then the code of underlying operation gets executed to perform that horizontal aggregation.

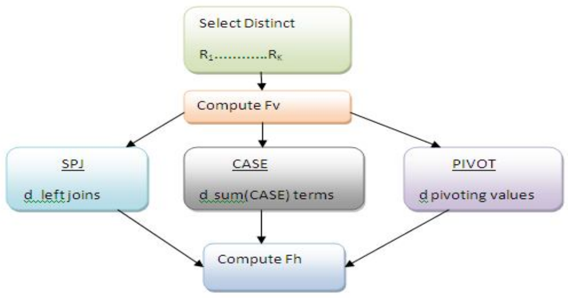

Fig. 3 shows steps on all methods based on table containing results of vertical aggregations

ISSN 2278-3091

All aggregation methods such as PIVOT, CASE, and SPJ proposed in this paper follow the steps as described in fig. 2. For each method the first step is issuing a SELECT query. Then other operator is applied as required. Then the code of underlying operation gets executed to perform that horizontal aggregation.

SPJ Method

Based on the relational operators the construct of this method is built. According to this method a table is created which exhibits vertical aggregation for every column. Afterwards, joining of all tables is performed in order to produce horizontal aggregations. The implementation of the SPJ method is based on the procedure given in [18].

PIVOT Method

Every RDBMS has a provision for PIVOT operation. The PIVOT operator that comes built in with RDBMS is used to build this construct programmatically. It can provide transpositions that can be utilized for evolution of resultant aggregations. The PIVOT operator knew the number of columns required to represent a transpose that can be combined with the operator GOUP BY of SQL.

```
SELECT L1,..,Lj
,sum(CASE WHEN R_1= v_11 and .. and R_k = v_k1
THEN A ELSE null END)
..
SELECT DISTINCT R_1
FROM F; /* produces v_1; . . . ; v_d */
SELECT
L_1; L_2; . . . ; L_j
,v_1; v_2; . . . ; v_d
INTO F_t
FROM F
SELECT L_1; L_2; . . . ; L_j;R_1, A
FROM F) F_t
PIVOT(
V (A) FOR R_1 in (v_1; v_2; . . . ; v_d)
) AS P;
```

Listing 1 – Shows elaboration instructions for PIVOT construct

As seen in listing 1, the optimized instructions are used in PIVOT construct. Only the columns that are required for horizontal aggregations are used for computations.

CASE Method

The existing CASE command of SQL is used to develop the CASE construct for horizontal aggregation. From the relational point of view it is very much similar to the aggregation/projection. It makes use of many conditions with conjunction. There are two strategies used for computing horizontal aggregations here. The first one is by using input table directly whereas the second one is by using vertical aggregation and store the resultant data into some arbitrary relation. That relation is further used to calculate horizontal aggregations. The implementation is based on the procedure given in [18].

EXPERIMENTAL EVALUATION

For development of the prototype application Visual Studio 2010 and SQL Server 2008 is the environment used. The experiments are done in a PC with 4 GB RAM, Core 2 Dual processor running Windows 7 operating system. ASP.NET and AJAX are the technologies used to build front end application with web based interface while the C# programming language is used for coding functionality. ADO.NET is used to interact with relational database. The result of SPJ construct for horizontal aggregation is presented in fig. 4.

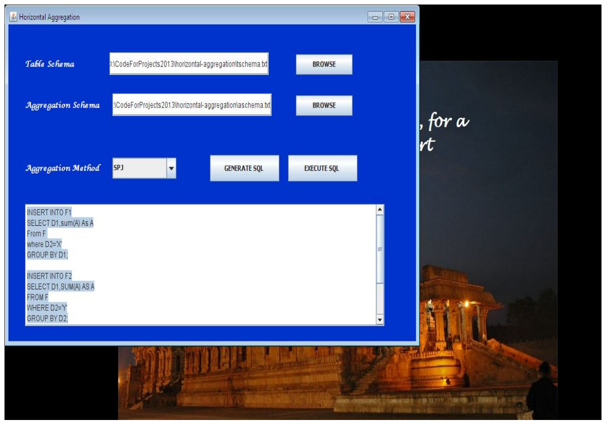

Table Schema:):\CodeForProjects2013\horizontal-aggregation\tschema.txt
Aggregation Schema: :\CodeForProjects2013\horizontal-aggregation\aschema.txt
Aggregation Method: SPJ

```
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 D2;
```

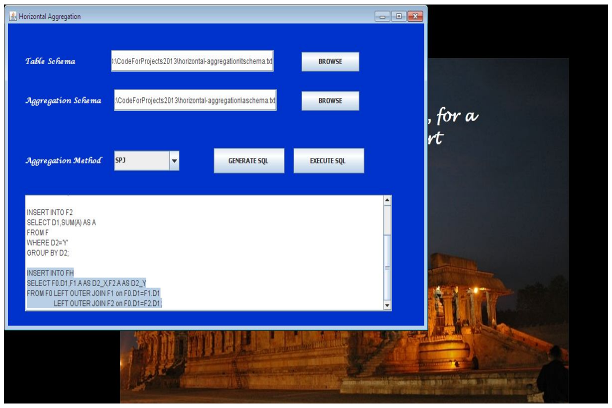

Table Schema:):\CodeForProjects2013\horizontal-aggregation\tschema.txt
Aggregation Schema: :\CodeForProjects2013\horizontal-aggregation\aschema.txt
Aggregation Method: SPJ

```
INSERT INTO F2
SELECT D1,SUM(A) AS A
FROM F
WHERE D2='Y'
GROUP BY D2;

INSERT INTO FH
SELECT F0.D1,F1.A AS D2_X,F2.A AS D2_Y
FROM F0 LEFT OUTER JOIN F1 on F0.D1=F1.D1
        LEFT OUTER JOIN F2 on F0.D1=F2.D1;
```

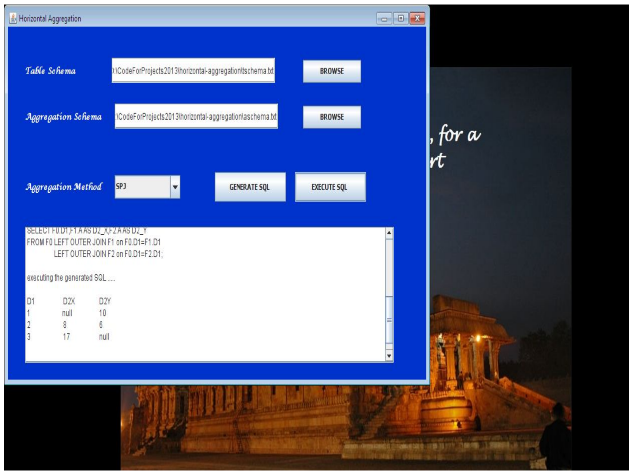

Table Schema: `):\CodeForProjects2013\horizontal-aggregation\tschema.txt`
Aggregation Schema: `:\CodeForProjects2013\horizontal-aggregation\aschema.txt`
Aggregation Method: SPJ

```
SELECT F0.D1,F1.A AS D2_X,F2.A AS D2_Y
FROM F0 LEFT OUTER JOIN F1 on F0.D1=F1.D1
          LEFT OUTER JOIN F2 on F0.D1=F2.D1;

executing the generated SQL .....

D1        D2X       D2Y
1         null      10
2         8         6
3         17        null
```

Fig. 4 – Results of SJP aggregation

As shown in fig. 4, the results of SPJ operator are presented in horizontal layout. The data in the resultant table can be used for data mining operations.

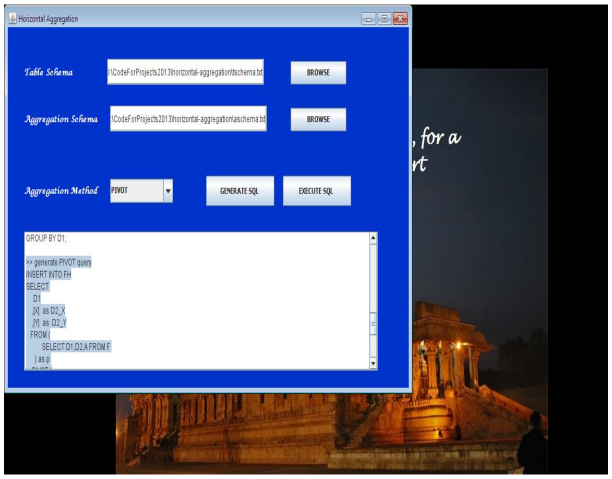

Table Schema: `):\CodeForProjects2013\horizontal-aggregation\tschema.txt`
Aggregation Schema: `:\CodeForProjects2013\horizontal-aggregation\aschema.txt`
Aggregation Method: PIVOT

```
GROUP BY D1;

>> generate PIVOT query
INSERT INTO FH
SELECT
  D1
  ,[X] as D2_X
  ,[Y] as D2_Y
 FROM (
      SELECT D1,D2,A FROM F
   ) as p
```

Fig. 5 – Result of Pivoting Aggregation

As seen in fig. 5, the results of PIVOT are presented in horizontal layout. The resultant data which is in tabular format can be used further for data mining.

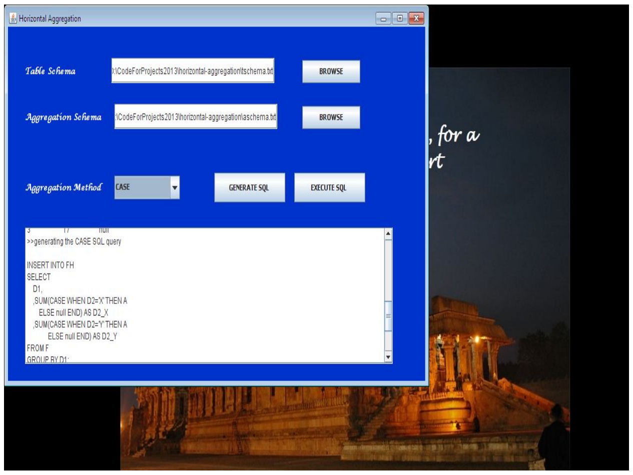

Table Schema: `):\CodeForProjects2013\horizontal-aggregation\tschema.txt`
Aggregation Schema: `:\CodeForProjects2013\horizontal-aggregation\aschema.txt`
Aggregation Method: CASE

```
3         17        null
>>generating the CASE SQL query

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;
```

Fig. 6 – Result of CASE Aggregation

As seen in fig. 6, the results of CASE operator are presented in horizontal layout. The data thus resulted can be used in data mining operations later. The aim of the operator is to generate datasets for data mining.

CONCLUSIONS

In this paper we built three constructs for horizontal aggregations of data. Thus it generates data sets for data mining. The provisions of SQL are used to build three aggregations namely PIVOT, CASE, and SPJ. It is achieved by using SQL as underlying query language. When these constructs are used by end user, the underlying commands are executed and the data sets are generated in horizontal layout. These operators can be used in OLAP applications where huge amount of historical data is analyzed. We built the operators for horizontal aggregations as the vertical aggregations supported by SQL return scalar values which are not suitable for data mining purposes. In order to build real world datasets for data mining we built the said operators. We built a web based prototype that demonstrates the proof of concept. The experimental results revealed that the proposed constructs for horizontal aggregations are effective in producing datasets for data mining.

REFERENCES

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

[2] C. Ordonez, "Statistical Model Computation with UDFs," IEEE Trans. Knowledge and Data Eng., vol. 22, no. 12, pp. 1752-1765, Dec. 2010.

[3] C. Ordonez and S. Pitchaimalai, "Bayesian Classifiers Programmed in SQL," IEEE Trans. Knowledge and Data Eng., vol. 22, no. 1, pp. 139-144, Jan. 2010.

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

[5] C. 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] S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications," Proc. ACM SIGMOD Int'l

Conf. Management of Data (SIGMOD '98), pp. 343-354, 1998.

[7] H. Wang, C. Zaniolo, and C.R. Luo, "ATLAS: A Small But Complete SQL Extension for Data Mining and Data Streams," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03), pp. 1113- 1116, 2003.

[8] A. Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, A. Gupta, L. Sheng, and S. Subramanian, "Spreadsheets in RDBMS for OLAP," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 52-63, 2003.

[9] H. Garcia-Molina, J.D. Ullman, and J. Widom, Database Systems: The Complete Book, first ed. Prentice Hall, 2001.

[10] C. Galindo-Legaria and A. Rosenthal, "Outer Join Simplification and Reordering for Query Optimization," ACM Trans. Database Systems, vol. 22, no. 1, pp. 43-73, 1997.

[11] G. Bhargava, P. Goel, and B.R. Iyer, "Hypergraph Based Reorderings of Outer Join Queries with Complex Predicates," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '95), pp. 304-315, 1995.

[12] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross- Tab and Sub-Total," Proc. Int'l Conf. Data Eng., pp. 152-159, 1996.

[13] G. Graefe, U. Fayyad, and S. Chaudhuri, "On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases," Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD '98), pp. 204-208, 1998.

[14] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman, "Non- Stop SQL/MX Primitives for Knowledge Discovery," Proc. ACM SIGKDD Fifth Int'l Conf. Knowledge Discovery and Data Mining (KDD '99), pp. 425-429, 1999.

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

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

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

[18] Carlos Ordonez and Zhibo Chen, "Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 4, APRIL 2012.

AUTHORS

Mr.Subbarao.Jasti received B.Tech in Information Technology from Bapatla Engineering College and pursing M.Tech in Computer Science from jawaharlal nehru technological university, kukatpally ,Hyderabad,AP, INDIA. His areas of interest includes Data Mining and Warehousing, Database Management Systems.

Prof. D Vasumathi received B.Tech in Computer Science and Engineering and Master's Degree M.Tech in Computer Science from JNT University Hyderabad, AP. She has been awarded Doctoral Degree (Ph.D) by the University of JNTU Hyderabad in computer Science. Currently she is working as an Professor in Department of Computer Science and Engineering in JNTU College of Engineering, kukatpally, Hyderabad, Andhra Pradesh. She has 12 years of Teaching and Administration experience in JNTU Hyderabad. She has published number of papers at national and international journals and conferences. She is a member of ISTE,IEEE,CSI et al. Her areas of interest includes Data Mining and Warehousing, Artificial Intelligence, Database Management Systems, Java and Web Technologies.