Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 8, August 2013, pg.230 – 236

RESEARCH ARTICLE

A Better Approach for Horizontal Aggregations in SQL Using Data Sets for Data Mining Analysis

Y. Chakravarthi[1], P. Vindhya[2]

[1]M. Tech, Dept. of CSE, ASIT, Gudur, India

[2]Assistant Professor, Dept. of CSE, ASIT, Gudur, India

[1] chakri.yannam@gmail.com, [2] vindhyachandranp@gmail.com

Abstract— To analyzing the data efficiently in Data mining systems are widely using datasets with columns in horizontal tabular layout. Generally preparing a data set is the more complex task in a data mining project, require many complex SQL queries, aggregating columns and joining tables. Conventional RDBMS usually manage tables with vertical form. Aggregated columns in a horizontal tabular layout returns set of numbers, instead of one number per row. This new class of function is called horizontal aggregations. The system uses one parent table and different child tables, operations are then performed on the data loaded from multiple tables. We proposed three fundamental methods .They are SPJ (select-project-join-Aggregation), CASE, and PIVOT. SPJ based on standard relational algebra operators. CASE is useful to exploiting the programming case construct. PIVOT is a built-in operator in a commercial DBMS, PIVOT operator, offered by RDBMS is used to calculate aggregate operations. PIVOT methods are much faster methods and offer much scalability. Partitioning large set of data, obtained from the result of horizontal aggregation.

Key Terms: - Aggregation; Data Mining; Data preparation; Structured Query Language (SQL); Pivot

I. INTRODUCTION

Horizontal aggregation is new class of function to return aggregated columns in a horizontal tabular layout. So many algorithms are required datasets with horizontal layout as input with several records and one variable per column. Managed large data sets without DBMS support can be a more difficult task. Different subsets of data points and dimensions are more flexible, easier and faster to do inside a relational database with SQL queries than outside with alternative tool. Horizontal aggregations can be performed by using operator; it is easily implemented inside a query processor, like a select, project and join operations. PIVOT operator on tabular data that exchange rows, enable data transformations useful in data modeling, data analysis, and data presentation. There are many existing functions and operators for aggregation in Structured Query Language.

The most commonly used aggregation is the sum of a column and other aggregation operators return the avg, maximum, minimum or row count over groups of rows. All operations for aggregation have many limitations to build large data sets for data mining purposes. Database schemas are also highly normalized for On -Line Trans action Processing (OLTP) systems where data sets that are stored in a relational database or data warehouse. But data mining, statistical algorithms generally require aggregated data in summarized form. Data mining algorithm requires suitable input in the form of cross tabular (horizontal) form; the significant effort is required to compute

© 2013, IJCSMC All Rights Reserved

aggregations for this purpose. Such effort is due to the amount and complexity of SQL code which needs to be written, optimized and tested. Data aggregation is a process in which information is gathered and expressed in a summary form, and which are used for purposes. A common aggregation purpose is to getting the more information about particular groups based on specific variables such as name, age, address, profession, phone number, or income. Most algorithms require input as a data set with a horizontal layout, with some records and one dimension or variable per column. That technique is used with models like clustering, classification, regression and PCA. Dimension used in data mining technique are point dimension.

II. RELATED WORK

SQL extensions are defining aggregate functions for association rule mining. Their optimizations have the purpose of avoiding the joins to convey unit (cell) formulas, but are not optimized to perform partial Transposition for each group of result rows. Conor Cunningalam [13] proposed an optimization and Execution strategies in an RDBMS which uses two operators i.e., PIVOT operator on tabular data that exchange rows and columns [1], enabled data transformations are useful in data modeling, data analysis, and data presentation. They can quite easily be implemented inside a query processor system, much like select, project, and join operator. Such design provides the opportunities for better performance, both during query optimization and query execution. Pivot is an extension of Group By with an unique restrictions and optimization opportunities, and this makes it is very simple to introduce incrementally on top of existing grouping implementations. H Wang.C.Zaniolo [2] proposed a s mall but Complete SQL Extension for Data Streams Data Mining and. This technique is a powerful database language and system that enables users develop complete data-intensive applications in SQL by writing new aggregates and table functions in SQL, rather than in procedural languages as in current Object -Relational systems .

The ATLaS system consist of applications including various data mining functions, that have been coded in ATLaS SQL, and execute with a modest (20– 40%) performance overhead with respect to the same applications written in C or C++ languages. This system can handle continuous queries using the schema and queries in Query Repository. Sarawagi, S.Thomas, R..Agrawal[3] proposed integrating association rule mining with relational database systems. The Integrating Association rule mining includes several methods. We first discuss research on extending SQL code for data mining processing. We compare horizontal aggregations with alternative proposals to perform pivoting. So many proposals are there to extended SQL syntax. The closest data mining problem associated to OLAP processing is association rule mining [3]. SQL extensions to define aggregate functions for association rule. Our SPJ method proved horizontal aggregations can be evaluated with relational algebra, exploiting the outer joins, showing our work is connected to traditional query optimization. The problems of optimizing queries with outer joins are not new. Optimizing joins by reordering operations and using transformation rules is studied in [14]. This work does not consider optimizing a complex query that contains several outer joins on primary keys only, which are basically to prepare data sets for data mining. Traditional query optimizers are used tree-based execution plan, but there is a work to facilitate advocates the use of hyper graphs to provide a more comprehensive set of potential plans. This approach is related to SPJ method. Even while the CASE construct is an SQL feature commonly used in practice optimizing queries that have a list of similar CASE statements has not been studied in depth before.

III. ADVANTAGES

In horizontal aggregations several are there. The first advantage is horizontal aggregation represent a template to generate SQL code from a data mining tool. This SQL code reduces manual work in the data preparation phase in data mining related project. The Second is automatically generating the SQL code, which is more efficient than an end user written SQL code. The datasets for the data mining projects can be created in less time. The third advantage is the data sets can be created entirely inside the DBMS.

IV. EXECUTIONS STRATEGIES IN HORIZONTAL AGGREGATIONS

Horizontal aggregations are proposed a new class of functions those aggregate the numeric expressions and the results are transposed to produce data sets with a horizontal layout. 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. This proposed system collect particular needed attributes from the different tables (fact tables) and displaying the columns in order to create data in the horizontal layout. The main goal of horizontal aggregation is to define a template to generate SQL code combining aggregation and pivoting (transposition). The second

© 2013, IJCSMC All Rights Reserved

goal is to extend the SELECT statement with a clause that combines the transposition with aggregation. We considering the following GROUP BY query in standard SQL that takes a subset $L_1,\ldots,L_m$ from $D_1,\ldots D_n$:

SELECT $L_1,\ldots,L_m$, sum (A) FROM F_1,F_2 GROUP BY L_1, L_m;

A. Input Parameters

There are four input parameters in horizontal aggregations. These parameters are generating SQL code:

1. The input table $F_1,F_2,\ldots,F_m$.
2. The list of GROUP BY columns $L_1,\ldots,L_m$.
3. The column is aggregate (A).
4. The list of transposing columns $R_1,\ldots R_k$.

That aggregation query will produce a wide table with m+1 columns (determined in automatically), with one group for each unique combination of values L_1,These aggregation query will be produces a wide table with m+1 columns (determined automatically), with one group for each unique combination of a values are $L_1,\ldots,L_m,\ldots..$, Lm and one aggregating value per group (sum(A)).In this order to evaluating the query these query optimizer taking the three input parameters. The first parameter is input table F(fact). The second one is the list of grouping columns $L_1,\ldots,L_m$. The third parameter is the column to aggregate (A).

B. Example

The bellow fig.1 shows there is a common field K in F and F_V, In F_V,D_2 consist of only two distinct values those are X and Y and these are used to transpose the table. Aggregate operation is used in this sum (). The values within D_1 are repeated,1 appears 3 times, for row 3, 4 and for row 3 and 4 values of D2 is X & Y. The D_2X and the D_2Y are newly generated columns in F_H.

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

D_1	D_2	A
1	X	Null
1	Y	10
2	X	8
2	Y	6
3	X	17

F_V

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

F_H

Fig.1: Example of Horizontal Aggregation

V. HORIZONTAL AGGREGATIONS

We introduce a new class of aggregations that contain related behavior to SQL standard aggregations, however which are produce tables with a horizontal layout. In contrast, we describe standard SQL aggregations vertical aggregations since they produce tables with a vertical layout. In Horizontal aggregations just they require a small syntax extension to aggregate functions that are called in a SELECT statement. Alternatively horizontal aggregations are used to generate SQL code from a data mining tool to build data sets for data mining analysis.

A. *SQL Code Generation*

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

© 2013, IJCSMC All Rights Reserved

transposition with aggregation. Consider the following GROUP BY query in standard SQL that takes a subset $L_1;\ldots;L_m$ from $D_1;\ldots;D_p$:

B. Proposed Syntax in Extended SQL

We must point out the proposed extension represents nonstandard SQL because the columns in the output table are not known when the query is parsed. We assume F does not change while a horizontal aggregation is evaluated because new values may create new result columns. Our main goal is to develop efficient evaluation mechanisms.

C. Example

We are using the above some rules and created horizontal table. Assume we want to summarize sales information with one store per row for one year sales. In more detail, we need the sales amount broken down by day of the week, the number of transactions by store per month, the number of items sold by department and total sales.

Table2
A Multidimensional Data Set in Horizontal Layout, Suitable for Data Mining

StoreId	SalesAmt			CountTransactions			Columns			sales
	Mon	Tue	… Sun	Jan	Feb	.. Dec	Dairy	meat	product	
10	125	141	140	2011	1807	4200	54	87	112	25054
30	80	98	88	802	912	1632	42	35	174	13876
.										
.										
.										

D. SQL Code Generation: Query Evaluation Methods

We propose three methods to evaluate horizontal aggregations. The first method is SPJ, it relies only on relational operations. That is, only doing select, join and aggregation queries; we call it the SPJ method. The second from relies on the SQL "case" construct; we call it the CASE method. Every table has an index on its primary key for efficient join processing. We don't consider additional indexing mechanisms to accelerate query evaluation. PIVOT operator is third method, It is a built-in operator which are transforms rows to columns (transposing). Figs .2 and 3 show an overview of the main steps to be explained below (for a sum() aggregation).

SPJ Method

The SPJ method is based on only relational operators. The fundamental concept in SPJ method is to build a table with vertical aggregation for each resultant column. To produce Horizontal aggregation F_H system must join all those tables. Two sub-strategies are compute Horizontal aggregation. The first strategy includes direct calculation of aggregation from fact table. Second one compute the corresponding the vertical aggregation and store it in temporary table F_V grouping by LE_1,….,LEi, RI_1,…..,RI_j then F_H can be computed from F_V.To get F_H system need n left outer join with n+1tables so that all individual aggregations are properly assembled as a set of n dimensions for each group. The Null value should be set as default value for groups with missing combinations for a particular group.

INSERT INTO F_H
SELECT $F_0.LE_1$, $F_0.LE_2$,….., $F_0.LE_j$,
F_1.A ,F_2.A ,……, F_n.A
FROM F_0
LEFT OUTER JOIN F_1
ON $F_0.LE_1 = F_1.LE_1$ AND … $F_0.LE_j = F_1.LE_j$
LEFT OUTER JOIN F_2
ON $F_0.LE_1 = F_2.LE_1$ AND …… $F_0.LE_j = F_2.LE_j$
LEFT OUTER JOIN F_n
ON $F_0.LE_1 = F_n.LE_1$ AND … $F_0.LE_j = F_n.LE_j$;

It is easy to see that left outer join is based on same columns. This strategy basically needs twice I/O operations by doing updates rather than insertion.

© 2013, IJCSMC All Rights Reserved

CASE Method

In SQL build-in "case" programming construct are available, it returns a selected value rather from a set of values based on Boolean expression. Queries for F_H can be evaluated by performing direct aggregation from fact table F and at the same time rows are transposing to produce the F_H.

SELECT DISTINCT RI_1
FROM F;
INSERT INTO F_H SELECT LE_1, LE_2,…..,LEj,
V(CASE WHEN RI1 = v_{11} and… R_k = vk_1 THEN A ELSE null
END) …..
, V(CASE WHEN RI_1 = $V1_m$ and… R_k= vk_1 THEN A ELSE null END)
FROM F
GROUP BY LE_1, LE_2, ,…, LE_j;

PIVOT Method

Pivot transforms a series of rows into a series of fewer of rows with additional columns [1]. Data in one source column is used to determine the new column for a row, and one more source column is used as the data for that new column. The wide form values can be consider as a matrix of column values, although the narrow form is a normal encoding of a sparse matrix.

In correct implementation PIVOT operator is used to calculate the aggregations. Individual method to express pivoting uses scalar sub queries, each one pivoted is created during a separate sub query. PIVOT operator provides a technique to allow rows to columns dynamically at the time of query compilation and execution. The example Query of PIVOT method is:

SELECT * FROM (Total bill Tab le PIVOT (SUM
(amount) for month in
(„Jan, Feb, Mar)) ;

This query will be generate atable with jan, feb and mar as column attribute and the sum of the amount of particular customer that are stored inside the bill Table. The pivot method is more efficient method compare to other two methods. Because the pivot operator is internally calculates the aggregation operation and no need to create extra tables. So operation performed with in this method is fewer compared to other methods.

The variation between the Optimized and UnOptimized will be show in the bellow tables

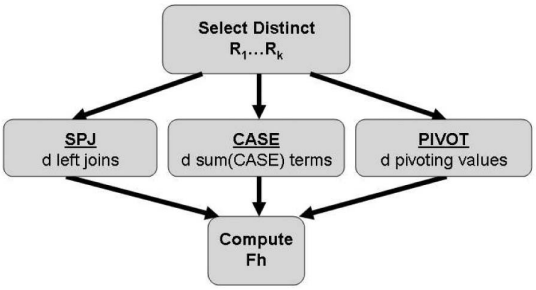

Fig. 2. The Main steps of methods based on F (unoptimized)

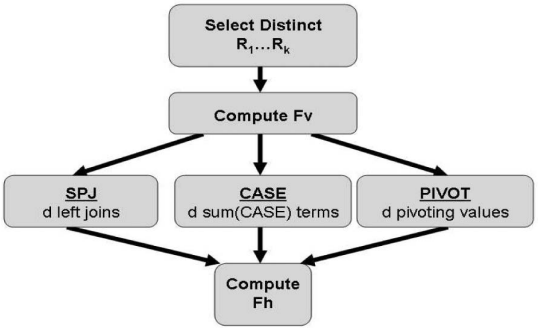

Fig. 3.The Main steps of methods based on FV (optimized)

© 2013, IJCSMC All Rights Reserved

VI. EXPERIMENTAL EVALUATION

In this experimental evaluation phase, we present the evaluation on a DBMS. We are evaluating the query optimizations. We compare the three query evaluation methods, and analyzing the time complexity varying table sizes and output data set dimensionality.

A. Setup: Computer Configuration and Data Sets

We are using the SQL Server V9, Dual Core Processor, running on a DBMS server running at 3.4 GHz, 5GB of RAM and 1 TB on disk. This SQL code generator is programming in the language of Java and connecting to the server via JDBC. The PIVOT operator is used as available in the SQL language implementation provided by commercial DBMS.

B. Query Optimization

Tables 2 analyze the first query optimization, applying to three methods. In this paper our goal was to assess the acceleration obtained by precomputing a cube and it storing on Fv. If we can see this optimization uniformly accelerating the all methods. this optimization is provide a different gain, depending on the method: the SPJ optimization is the best for small n, for PIVOT is large n, and the CASE there is rather a less dramatic improvement the all across n. The PIVOT is accelerated by our optimization despite the fact it is handled by the query optimizer. Since this optimization can produce a significant acceleration for this three methods (at least 2 x faster) we will use by default. We noticed the precomputing Fv taking the same time within each method.

TABLE 2
Query Optimization: Precompute Vertical Aggregation in F_v (N= 12M). Times in Seconds

n	d	SPJ		PIVOT		CASE	
		F	F_v	F	F_v	F	F_v
1K	7	552	62	121	58	119	59
	12	845	67	122	62	122	64
	25	1619	68	145	61	153	61
100K	7	540	86	131	81	247	81
	12	856	85	136	77	234	80
	25	1633	103	172	89	377	92
1.5M	7	669	230	538	157	242	155
	12	1051	419	751	140	322	141
	25	2776	1086	1240	161	384	150

C. Comparing Evaluation Methods

Table 3 is comparing the three query optimization methods. Notice that Table 3 is a Summarized version of the Table 2 shows the best time for each method.

TABLE 3
Comparing Query Evaluation Methods (All with Optimization computing Fv) Times in Seconds

N	n	d	SPJ	PIVOT	CASE
12M	1K	7	62	58	59
		12	67	62	64
		25	68	61	61
12M	100K	7	86	81	81
		12	85	77	80
		25	103	89	92
12M	1.5M	7	230	157	155
		12	419	140	141
		25	1086	161	150

VII. CONCLUSION

We are introduced a new class of extended aggregate functions, called a horizontal aggregations. Which are help to preparing datasets for OLAP cube exploration and data mining. In particularly, horizontal aggregations are useful to create data sets with a horizontal layout. Mainly a horizontal aggregation returns a set of numbers instead of one number per each group. For a query optimization perspective, we are proposed the three

© 2013, IJCSMC All Rights Reserved

fundamental query evaluation methods. The first method is SPJ. It relies on standard relational operators. The second is CASE. It relies on the case construct. It shows the indexes of the tables. PIVOT is the last method. The pivot is a built in operator. Generally PIVOT operator is shows the table in two ways (Narrow, wide tables). It is a built-in operator in a commercial database. SPJ method is important from a theoretical point of view because it is based on select, project and join queries. Both CASE and PIVOT evaluation methods are significantly faster than the SPJ method.

REFERENCES

[1] C. Cunningham, G. Graefe, and C. A. Galinda, "PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS," proc. 13th int'l conf. Very Large Data Base (VLDB '04), pp, 998-1009,2004.

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

[3] S.Sarawagi, S.Thomas, and R. Agrawal, "Integrating Association Rule Mining with Relational Database System: Alternatives and Implications," proc. ACM SIGMOD Int'l Conf. Management of ata (ZIGMOD'98), pp. 343-354, 1998.

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

[5] J. Gray, A. Bosworth, A. Layman and H. Pirahesh. A Relational Aggregation Operator Generalizing Group-By ,Cross-Tab and Sub Total. In ICDE Conference,pages 152-159, 1996.

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

[7] J. Luo, J.F. Naughton, C.J, Ellmann, and M .Watzke. Locking Protocols for Materialized Aggregate Join Views. IEEE Transactions on Knowledge and Data Engineering (TKDE), 17 (6): 796-807, 2005.

[8] C.Ordonez and S. Pitchaimalai. Batesian Classifiers Programmed in SQL.IEEE Transactions on Knowledge and Data Engineering (TKDE),22(1): 139-144, 2010.

[9] C. Ordonez and Zhibo Chen, "Horizontal Aggregation in SQL to Prepare Data Sets for Data Mining Analysis," IEEE Trans. On Knowledge and Data Engineering (TKDE), 1041-4347/$26.00, 2011.

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

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

[12] 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

© 2013, IJCSMC All Rights Reserved