ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

A Literature Review on Horizontal Aggregations in SQL to Prepare Data Sets for Data Mining Analysis

Sanjay Gandhi G[1], Dr.Balaji S[2]

Associate Professor, Dept. of CSE, VISIT Engg College, Tadepalligudem, Scholar Bangalore University, India[1]

Professor, Dept. of CSE, KL University, Andhra Pradesh, India[2]

ABSTRACT: Now days due to popular of E-Commerce so much data is generating and from these data preparing dataset will take more time or tedious task to preparing queries and generating datasets. According to Data mining analysis because data mining discipline required to write more and more complex SQL queries and so many tables are to be joined to get the aggregated result. In literature review SQL aggregations organize the data sets in vertical draft that is they return result on one column per aggregated group. But in the data mining, the data set to be required in horizontal layout. So to overcome such problems transform the data into suitable dataset form the existing three horizontal aggregation methods are used. In this paper we are proposing three different horizontal aggregation are SPJ (select, project, join) method, CASE method and PIVOT method. The study become more capable if the dataset achieve is in the horizontal form. The main goal is to identify the most proficient method from these three methods in terms of time and space complexity. So these methods are compared using large tables and identified that the CASE method is more efficient than SPJ and PIVOT method. In general, the CASE and PIVOT methods exhibit linear scalability, whereas the SPJ method does not.

KEYWORDS: Aggregate functions, Data Set Preparation, Pivoting; SQL

I. INTRODUCTION

Data mining refers to the finding of relevant and useful information from databases. A data mining project consists of several phases. The first phase is called the data preparation phase. The second phase involves the analyzing of data sets using data mining algorithms. The third phase involves validating of results. The fourth phase involves deploying of statistical results on new data sets. The first phase involves the extracting data from multiple operational databases and from external sources, cleaning of data where unnecessary information is removed, and transforming data into suitable form for the task of data mining. For transforming the data, the aggregation in SQL is used. In SQL, the aggregation of data is done using the aggregate functions such as minimum, maximum, average, count and sum and the result is obtained in the vertical layout. By using this data set as such, the person that done data miners need to write large SQL queries to convert it into the appropriate form.

Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout instead of a single value per row. Horizontal aggregations provide several unique features and advantages. First, they represent a template to generate SQL code from a data mining tool. 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. Third, the data set can be created entirely inside the DBMS. 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. To perform horizontal aggregation, SPJ method is implemented with generalized Projections. GPs capture aggregations, group- conventional projection with duplicate elimination (distinct), and duplicate preserving projections. We develop a technique for pushing GPs down query trees of Select project-join may use aggregations like max, sum, etc. and that use arbitrary functions in their selection conditions. Our technique pushes down to the lowest levels of a query tree aggregation computation, duplicate elimination, and function computation.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

Existing SQL aggregate functions present important limitations to compute percentages. This article proposes two SQL aggregate functions to compute percentages addressing such limitations. The first function returns one row for each percentage in vertical form like standard SQL aggregations. The second function returns each set of percentages adding 100% on the same row in horizontal form. These novel aggregate functions are used as a framework to introduce the concept of percentage queries and to generate efficient SQL code. Experiments study different percentage query optimization strategies and compare evaluation time of percentage queries taking advantage of our proposed aggregations against queries using available OLAP extensions. The proposed percentage aggregations are easy to use, have wide applicability and can be efficiently evaluated.

II. LITERATURE REVIEW

Authors: E.F. Codd. “Extending the Database Relational Model to Capture More Meaning”
During the last three or four years several investigators have been exploring “semantic models ” for formatted databases. The intent is to capture (in a more or less formal way) more of the meaning of the data so that database design can become more systematic and the database system itself can behave more intelligently. Two major thrusts are clear: (I) the search for meaningful units that are as small as possible--atomic semantics; (2) the search for meaningful units that are larger than the usual n-ary relation-molecular semantics. In this paper we propose extensions to the relational model to support certain atomic and molecular semantics. These extensions represent a synthesis of many ideas from the published work in semantic modeling plus the introduction of new rules for insertion, update, and deletion, as well as new algebraic operators.
In this author proposed extensions to the relational model to support certain atomic and molecular semantics. These extensions represent a synthesis of many ideas from the published work in semantic modeling plus the introduction of new rules for insertion, update, and deletion, as well as new algebraic operators.

Authors: C. Cunningham, G. Graefe, and C.A. Galindo-Legaria “PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS”
PIVOT and UNPIVOT, two operators on tabular data that exchange rows and columns, enable data transformations useful in data modeling, data analysis, and data presentation. 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. We discuss query optimization and execution implications of this integrated design and evaluate the performance of this approach using a prototype implementation in Microsoft SQL Server.

Authors: C. Galindo-Legaria and A. Rosenthal. **“**Outer join simplification and reordering for query optimization.**”**

Conventional database optimizers take full advantage of associatively and commutatively properties of join to implement efficient and powerful optimizations on select/project/join queries. However, only limited optimization is performed on other binary operators. In this article, we present the theory and algorithms needed to generate alternative evaluation orders for the optimization of queries containing outerjoins. Our results include both a complete set of transformation rules, suitable for new-generation, transformation-based optimizers, and a bottom-up join enumeration algorithm compatible with those used by traditional optimizers.

Authors: C. Ordonez. “Vertical and horizontal percentage aggregations.**”**

Existing SQL aggregate functions present important limitations to compute percentages. This article proposes two SQL aggregate functions to compute percentages addressing such limitations. The first function returns one row for each percentage in vertical form like standard SQL aggregations. The second function returns each set of percentages adding 100% on the same row in horizontal form. These novel aggregate functions are used as a framework to introduce the concept of percentage queries and to generate efficient SQL code. Experiments study different percentage query optimization strategies and compare evaluation time of percentage queries taking advantage of our proposed aggregations against queries using available OLAP extensions. The proposed percentage aggregations are easy to use, have wide applicability and can be efficiently evaluated.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

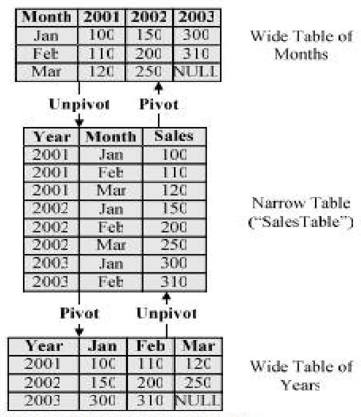
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

III. PIVOT AND UNPIVOT IN SQL

It is possible to implement pivoting in standard SQL, though the syntax is cumbersome and its performance is generally poor. One method to express pivoting uses scalar subqueries in the projection list. Each pivoted column is created through a separate (but nearly identical) subquery as seen in Figure 3. For database implementations that do not support PIVOT, users could employ this technique to perform pivoting operations.



Month	2001	2002	2003
Jan	100	150	300
Feb	110	200	310
Mar	120	250	NULL

Year	Month	Sales
2001	Jan	100
2001	Feb	110
2001	Mar	120
2002	Jan	150
2002	Feb	200
2002	Mar	250
2003	Jan	300
2003	Feb	310

Year	Jan	Feb	Mar
2001	100	110	120
2002	150	200	250
2003	300	310	NULL

Figure 1 Pivot and Unpivot

Inclusion of Pivot and Unpivot inside the RDBMS enables interesting and useful possibilities for data modeling. Existing modeling techniques must decide both the relationships between tables and the attributes within those tables to persist. The requirement that columns be strongly defined contrasts with the nature of rows, which can be added and removed easily. Pivot and Unpivot, which exchange the role of rows and columns, allow the a priori requirement for pre-defined columns to be relaxed. These operators provide a technique to allow rows to become columns dynamically at the time of query compilation and execution. When the set of columns cannot be determined in advance, one common table design scenario employs "property tables", where a table containing (id, propertyname, propertyvalue) is used to store a series of values in rows that would be desirable to represent columns. Users typically use this design to avoid RDBMS implementation restrictions (such as an upper limit for the number of columns in a table or storage overhead associated with many empty columns in a row) or to avoid changing the schema when a new property needs to be added. This design choice has implications on how tables in this form can be used and how well they perform in queries. Property table queries are more difficult to write and maintain, and the complexity of the operation may result in less optimal query execution plans. In general, applications written to handle data stored in property tables cannot easily process data in the wide (pivoted) format. Pivot and Unpivot enable property tables to look like regular tables (and vice versa) to a data modeling tool. These operations provide the framework to enable useful extensions to data modeling.

IV. EXECUTION STRATEGIES

Defining PIVOT in terms of GROUP BY and Apply provide an excellent opportunity to re-use existing execution operators in new ways. In Section 3.3, we demonstrated that PIVOT can be implemented as GROUP BY. Hash and stream aggregation are available for PIVOT, and have similar execution properties. Parallel query execution can also be supported using these execution strategies as long as the members of each group are processed in the same thread. PIVOT does use a relatively large number of identical aggregates with almost identical scalar logic. One novel execution strategy could group the computation of these aggregates together, either by treating the set of aggregates as a vector computation or by rewriting each individual aggregate computation into a dispatch table (as each column will be looking for a single and likely unique scalar for each input row).

PIVOT can also be implemented through a specialpurpose iterator transposing rows into columns. Consuming a sorted (grouping columns and the pivot column) stream, the next row in the current group becomes the source of the value for

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

the next column. If a pivoted column does not have a corresponding row in the input, it returns the empty value for all output columns until the correct location for the current input row is located. Similar to the grouping operators, this technique can be performed simultaneously over values from different groups.
As described in this paper, UNPIVOT can be implemented as a correlated nested loops join (Apply). Each invocation of the Apply can be performed in parallel, leveraging existing parallel techniques available to joins. UNPIVOT can also be implemented using a special purpose execution iterator that consumes one row and returns a number of rows in unpivoted form. Parallelism is slightly easier for UNPIVOT since each input row can be processed independently (instead of groups of rows).

V. CONCLUSION

In this paper we reviewed on two new data manipulation operators, Pivot and Unpivot, for use inside the RDBMS. These reviews will study many existing user scenarios and enable several new ones. Furthermore, this paper outlines the basic syntactic, semantic, and implementation issues necessary to add this functionality to an existing RDBMS based on algebraic, cost-based optimization and algebraic data flow execution. Pivot is an extension of Group By with unique restrictions and optimization opportunities, and this makes it very easy to introduce incrementally on top of existing grouping implementations. Finally, we present a number of axioms of algebraic transformations useful in an implementation of Pivot and Unpivot.

REFERECES

[1] A. Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, A. Gupta, L. Sheng, and S. Subramanian. Spreadsheets in RDBMS for OLAP. In Proc. ACM SIGMOD Coference, pages 52.63, 2003.
[2] Venky Harinarayan ,Ashish Guptay "Generalized Projections: a Powerful Query-Optimization Technique "
[3] "Vertical and Horizontal Percentage Aggregations", Carlos Ordonez Teradata, NCR San Diego, CA 92127, USA.
[4.] C. Ordonez. Integrating K-means clustering with a relational DBMS using SQL. IEEE Transactions on Knowledge and Data Engineering (TKDE), 18(2):188–201, 2006.
[5.] G. Graefe, U. Fayyad, and S. Chaudhuri. On the efficient gathering of sufficient statistics for classification from large SQL databases. In Proc. ACM KDD Conference, pages 204–208, 1998.
[6.] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and subtotal. In ICDE Conference, pages 152–159,1996.
[7.] G. 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. Bayesian classifiers programmed in SQL. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22(1):139–144, 2010.
[9] C. Ordonez. Horizontal aggregations for building tabular data sets. In Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop, pages 35.42, 2004.
[10] C. Ordonez, Zhibo Chen. Horizontal aggregations in SQL to prepare Data Sets for Data Mining Analysis. IEEE Transactions on Knowledge and Data Engineering (TKDE), 2012.
[11.] C. Ordonez. Vertical and horizontal percentage aggregations. In Proc. ACM IGMOD Conference, pages 866–871, 2004.
[12.] C. C. Ordonez and Zhibo Chen. Horizontal Aggregation in SQL to prepare Data Sets forData Mining Analysis. . IEEE Transactions on Knowledge and Data Engineering (TKDE), 1041- 4347/11/$26.00 ,2011
[13] C. Ordonez, "Vertical and horizontal percentage aggregations", . In Proc. ACM SIGMOD Conference, pages 866–871, 2004.
[14] C. Ordonez, "Integrating K-means clustering with a relational DBMS using SQL", IEEE Transactions on Knowledge and Data Engineering (TKDE), 18(2):188–201, 2006.
[15] C. Ordonez, " Statistical model computation with UDFs", IEEE Transac-tions on Knowledge and Data Engineering (TKDE), 22, 2010.