International Journal of Computer Science
and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 3, July 2013, 27-34

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

PERFORMANCE EVALUATION OF HORIZONTAL AGGREGATION TECHNIQUES IN SQL

LAVINA D. PANJWANI & RICHA K. MAKHIJANI

Department of Computer Science and Engineering, S. S. G. B. C. O. E. T., Bhusawal, Maharashtra, India

ABSTRACT

Data mining (DM) can be viewed as a result of the natural evolution of information technology. For DM, data needs to be transferred into a data-mining-capable format**.** From an assortment of methods of data transformation, one form is horizontal aggregation methods. Three methods of horizontal aggregation used in proposed methodology are SPJ, CASE and PIVOT. These methods generate horizontally aggregated datasets. Evaluating these methods and their performance led to several interesting minutiae which in juxtaposition with experimental results are bequeathed.

General Terms: SQL

KEYWORDS: Horizontal Aggregation, SPJ, CASE, PIVOT, Primary Index, Secondary Index

INTRODUCTION

Wide availability of huge amounts of data have led to looming necessitate for transforming such data into useful information and knowledge. This escorted to knowledge management which led to an intricate and iterative process called Knowledge Discovery in Databases (KDD).

Knowledge Discovery in Databases

KDD is a complex process concerned with the discovery of relationships and other descriptions from data. The approach to gain knowledge out of a set of data was separated by Fayyad into individual steps [2] as shown in figure 1. According to Fayyad there are following steps: Selection, Pre-processing, Transformation, Data Mining and Interpretation. In the *Selection*-step the significant data gets selected or created. Henceforward the KDD process is maintained on the gathered target data. Important elements of the provided data have to be detected and filtered out. These kinds of things are settled in the *Pre-processing* phase. The *Transformation* phase of the data may result in a number of different data formats, since variable data mining tools may require variable formats. The data also is manually or automatically reduced. In the *Data Mining* phase, the data mining task is approached. The *Interpretation* of the detected pattern reveals whether or not the pattern is interesting. That is, whether they contain knowledge at all. This is why this step is also called evaluation.

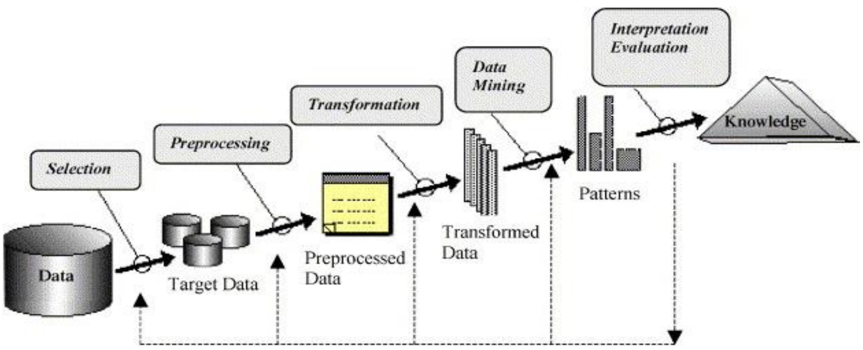

Figure 1: An Overview of Steps in Knowledge Discovery in Databases

Data Transformation

A good result after applying data mining depends on an appropriate data preparation and transformation in the beginning. A representative selection can be used to draw conclusions to the entire data. Data transformation processes the input data and changes the representation of the input such that the resulting output enlightens more features and avails supplementary possibility for data mining.

Organization of Paper

In this paper, section 2 elaborates literature survey. Section 3 gives an insightful of proposed methodology and architecture. Section 4 is performance evaluation which enlightens methods of evaluation. Section 5 provides experimental results and section 6 is conclusion.

LITERATURE SURVEY

Today data is stored in data warehouses which motivate analytics to perform investigative illustration and bring out hidden acquaintance for explorations. For this revelation, data needs to be transformed from their original crude condition into a new form, or representation that is suitable for utilization. In spite of many characteristics of data like data type, level of structure etc. some of data transformations and data representations techniques are conferred from ‘Illuminating the Path’. Dimensionality reduction techniques provide generalized methods for data simplification [7].

Researchers conveyed that reduction in dimensionality could be possible if either number of variables is decreased or scrutiny to be managed is trimmed down. These basic principles form base for some schemes: Principal components analysis (PCA), Multi-Dimensional Scaling (MDS), Clustering and more [7].

In PCA, new variables are produced by combinations of original variables whereas in MDS small sized pseudo-vectors are created that approximate the high dimensional structure of data in a lower-dimensionality representation. Clustering of homogeneous data is also effective method for reducing the number of observations to be managed.

Carlos Ordonez has proposed two methods to overcome limitation of SQL to compute percentages. The first function, called vertical percentage returns one row for each percentage in vertical form like standard SQL aggregations. The second function, called horizontal percentage returns each set of percentages adding 100% on the same row in horizontal form. Queries using percentage aggregations are called percentage queries. These aggregate functions were used as a framework to introduce the concept of percentage queries and to generate efficient SQL code [3].

C. Ordonez introduced a technique to efficiently compute fundamental statistical models inside a DBMS exploiting User-Defined Functions (UDFs). Two layouts for the input data set: horizontal and vertical, are considered. Authors have introduced efficient SQL queries to compute summary matrices and score the data set.

Based on the SQL framework, later introduced UDFs that worked in a single table scan: aggregate UDFs to compute summary matrices for all models and a set of primitive scalar UDFs to score data sets [4].

PROPOSED METHODOLOGY

Data transformation is done prior to DM to achieve better results. The proposed framework comprises of importing databases, implementation of methods perform data transformation and generate horizontally aggregated datasets, saving the results, allows generation of mini-datasets and evaluation of methods. The proposed system architecture is as shown in figure 2.

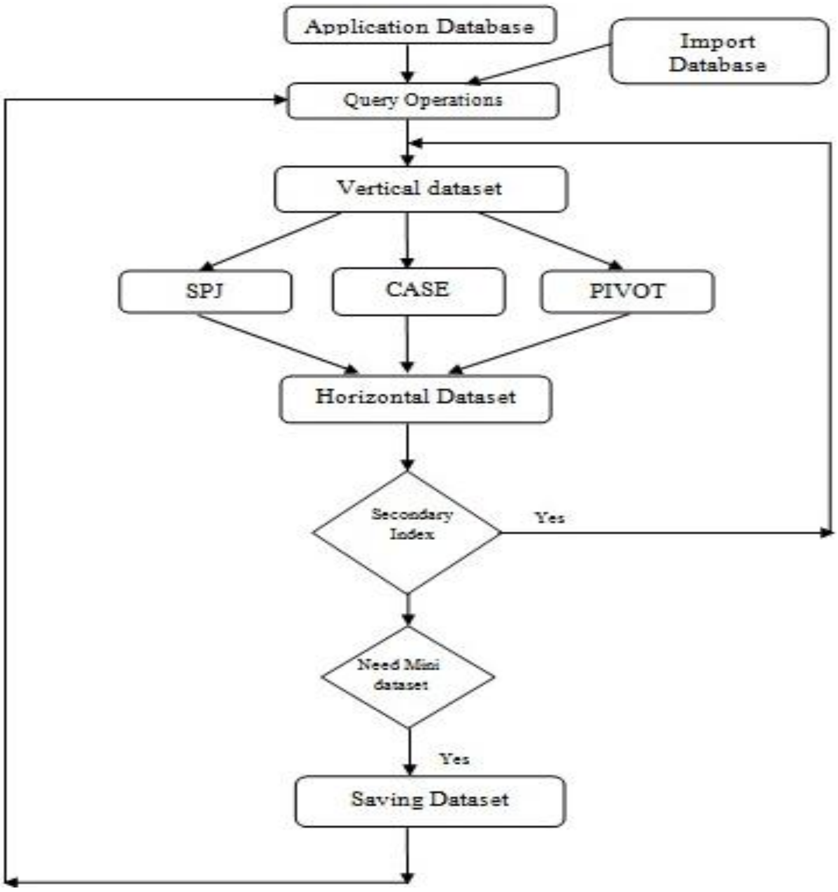

Figure 2: Proposed Methodology

The methods implemented are SPJ, CASE and PIVOT. All the three methods generate datasets that are aggregated and are in horizontal (de-normalized) form. First method is SPJ, which allows select and join (using clause) operations to be projected along with aggregation function. CASE method used 'CASE' construct of SQL and allows grouping of the information in output dataset. PIVOT makes use of 'pivot' operator for output generation. As the proposed framework incorporates 'Import database' facility, this makes system dynamic and not restricted to just one database. Database may be MS-Access or SQL File based or SQL Server based, all three kinds of databases can be integrated into the system. User may also save in the output which is in horizontally aggregated form, to be saved in as a new relation in the same database which was the input. User only needs to provide relation name and click, auto query generation is transpired. Once user has saved in the dataset as a relation, she can perform various DML operations of SQL to generate mini-datasets, i.e. diminutive version of the input dataset. Mini-datasets allows end user to explore selective section of huge dataset precisely. In next section, performance evaluation is discussed.

PERFORMANCE EVALUATION

SPJ, CASE and PIVOT are methods that are used to generate horizontally aggregated datasets in the proposed architecture. In first method, SPJ stands for Select, Project and Join with a clause and aggregation. SQL provides many join operations but in SPJ, joining of two or more relation is done using clause which is accompaniment by means of conditions. In next method, CASE construct is used for transformation and generation of datasets. Using CASE, allows the user to classify the input data into category or groups. This helps in exploiting the data, in terms of class and titles. In PIVOT method, in-built SQL operator 'pivot' is used. Using pivot operator allows transposing the input relation directly. All the three methods are proficient of generating same datasets as output provided same input along with same conditions are used. Performance of all the three methods is evaluated by observing time taken by each method to generate horizontally aggregated dataset for various inputs. On the whole, the evaluation leads to specifics listed: as the number of rows and columns increase, time taken by methods gradually increase; SPJ and CASE consume more time as compared to PIVOT; most imperative is that as input change (due to import database feature) the SQL queries also vary for each of the given methods which impinge on the recital of all three input methods.

EXPERIMENTAL RESULTS

In this section, experimental setup, results and observations are conversed and exhibited.

Setup

SQL Server 2005 is used running on a system with Intel Core i-3 processor at 2.3 GHz and 4GB RAM. A code generator is designed .Net framework in Visual C#.

Results

The three methods viz. SPJ, CASE and PIVOT of data transformation were executed and time intervened for generating horizontally aggregated datasets was noted for performance evaluation of these methods. Basic notations of the tables are as stated: n – number of rows in opted database/ relation; d- number of distinct columns in the opted database/ relation; DB is the input database which is un-optimized; Tab-V indicates the optimized input for data transformation; Tab-H denotes the output i.e. horizontally aggregated columns driven from Tab-V. Every reading of any of the given tables is calculated by taking the average of five readings of respective event/ method. Figure 3 and 4 flaunt the proposed framework in execution.

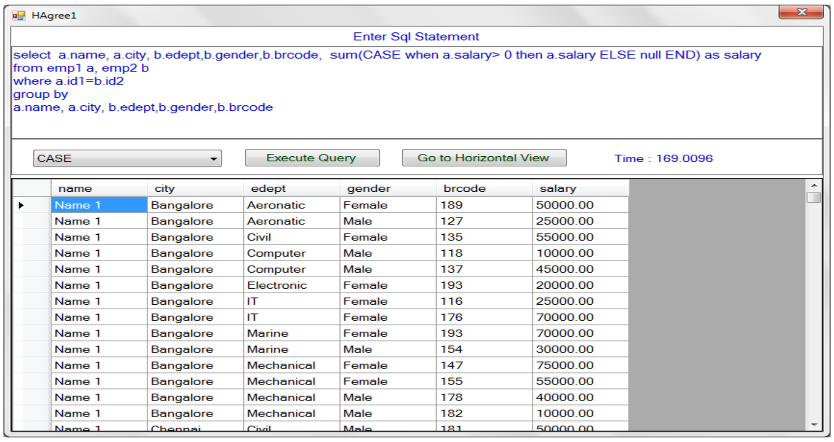

Figure 3: Horizontal Aggregation and Query Execution

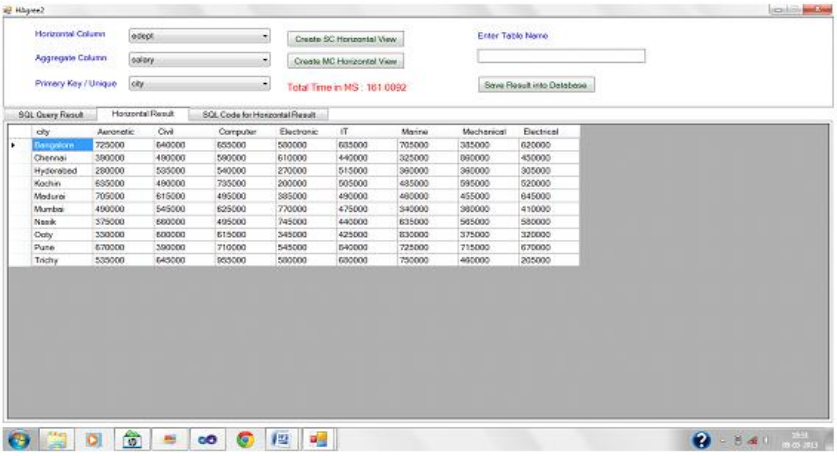

Figure 4: Horizontally Aggregated Dataset Generation

Using the proposed framework, helps in achieving query optimization as Tab-H is generated from Tab-V which is optimized. Table 1 displays difference between times elapsed in generating DB and Tab-V for each of the three methods. Notion primary index refers to any column from the considered relation/ database that have a unique value for every entity. Notion secondary index refers to any column from the considered relation/ database that may have several distinct values but not a sole value for every entity. Table 2 points out the times taken by three methods to generate Tab-H for single relation taken primary and secondary index into consideration. Table 3 points out the times taken by three methods to generate Tab-H for multiple relations taken primary and secondary index into consideration.

Table 1: Query Optimization – Exhibiting Time Needed to Generate DB (Un-Optimized) and Tab-V (Optimized), where Time is in Milli-Seconds

n	d	SPJ		CASE		PIVOT	
		DB	Tab-V	DB	Tab-V	DB	Tab-V
1K	5	30	28	52	45	40.6	23
	8	59	44	49	36	50.4	35.4
	10	82	54	69	56	44.8	42
2K	5	51	48	87	47	53.2	25.4
	8	69	56	59	42	63.8	55.6
	10	98	69	84	65	70	56.2
4K	5	72	68	97	75	68.2	66.2
	8	92	82	95	83	88	77
	10	103	96	131	99	103	94.8

Table 2: Performance Evaluation of Methods (Primary and Secondary Index) for Single Relation – Exhibiting Time Needed to Generate Horizontal Aggregated Datasets, where Time is in Seconds

n	d	SPJ		CASE		PIVOT	
		Primary Index	Secondary Index	Primary Index	Secondary Index	Primary Index	Secondary Index
1K	5	7.91	0.12	7.82	0.112	0.018	0.018
	8	7.81	0.14	7.92	0.170	0.021	0.022
	10	7.93	0.41	7.99	0.156	0.045	0.045
2K	5	32.22	0.15	30.34	0.143	0.025	0.025
	8	30.30	0.24	30.83	0.252	0.031	0.031
	10	30.69	0.26	30.45	0.237	0.068	0.068
4K	5	127.62	0.17	119.96	0.168	0.046	0.046
	8	120.61	0.41	121.16	0.423	0.053	0.053
	10	121.20	0.41	121.46	0.411	0.099	0.099

Table 3: Performance Evaluation of Methods (Primary and Secondary Index) for Multiple Relations – Exhibiting Time Needed to Generate Horizontal Aggregated Datasets, where Time is in Seconds

n	d	SPJ		CASE		PIVOT	
		Primary Index	Secondary Index	Primary Index	Secondary Index	Primary Index	Secondary Index
1K	9	7.97	0.121	7.96	0.15	0.031	0.049
	11	7.88	0.128	7.93	0.15	0.059	0.043
2K	9	32.33	0.257	30.63	0.25	0.056	0.053
	11	30.49	0.225	38.20	0.26	0.056	0.052
4K	9	126.02	0.469	120.66	0.41	0.221	0.074
	11	120.33	0.386	120.25	0.43	0.187	0.078

Results Observation

SPJ, CASE and PIVOT methods for data transformation provide the provision to generate the datasets with horizontal aggregation i.e. new columns which originally did not exist in input. Aggregation is performed for these newly generated columns along with the existing columns. Evaluating these methods led to some interesting facts. As different values of n and d were taken into consideration, it was visibly pragmatic that as the value on n and d increased, time taken to generate Tab-H also increased. Comparing the three methods led to essentials that SPJ and CASE take just about same time for Tab-H for small value of n and d. Also with increase in the values of d and n, SPJ takes more time than CASE. Apart from SPJ and CASE, PIVOT take significantly less time to generate Tab-H for both small and immense values of n and d. Also in all the three methods, time taken to generate Tab-H for primary key/ index is too hefty than that taken for secondary index. Figure 5 and 6 display the graphical representation of the values from table 2 (Row: n=2k ; d=8) for primary and secondary index respectively.

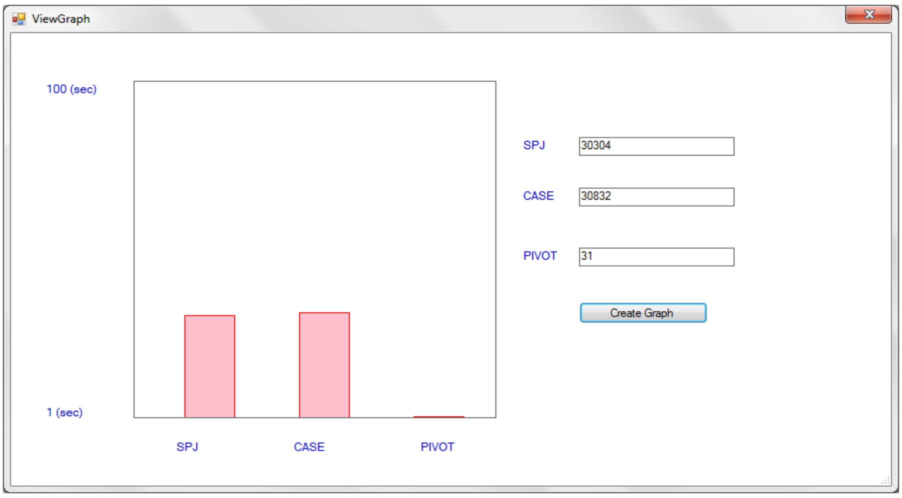

Figure 5: Graph Putting on View Time Needed by SPJ, CASE and PIVOT Method for Primary Index, where Input is Time Milli-Sec and Output Displays Time in Seconds

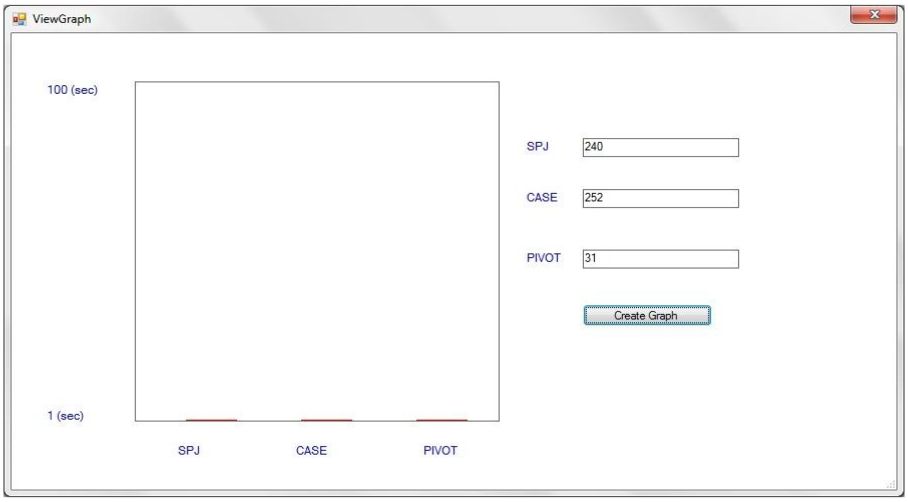

Figure 6: Graph Putting on View Time Needed by SPJ, CASE and PIVOT Method for Secondary Index, where Input is Time Milli-Sec and Output Displays Time in Seconds

CONCLUSIONS

Data transformation is an imperative stage of knowledge discovery process. Data transformation yields output which is considered as a better input for data mining process. SQL provides aggregation function which generates output as a single row and no increase in number of columns. The three methods demonstrated in proposed framework SPJ, CASE and PIVOT, overcome these shortcomings of SQL vertical aggregation. SPJ and CASE methods are traditionally not in-built function/ feature of SQL to generate horizontal aggregated columns; whereas PIVOT uses an in-built feature of SQL in grouping along aggregation function. Out of three methods from the framework, SPJ and CASE methods allow drop down option for available column name to generate new column whereas in PIVOT each new column name is provided as part of SQL query. Fundamentally PIVOT generates output in least possible time and can thought of as a fastest method out of three. But the implementation of SPJ and CASE in this framework reduces complexity of writing SQL query for the user. In essence, key observation legitimate for all the three methods is that, output for secondary index was generated in much diminutive time as that for primary index. For future work, more methods for data transformation and horizontal aggregation can be brought into contemplation. Also, these proposed framework and methods can be weathered for databases engendered by the TPC-H generator.

REFERENCES

1. ”Data Mining: Concepts and Techniques”, Jiawei Han and Micheline Kamber, 1999, ch. 1, Page 3 – 21 at URL http://db.cs.sfu.ca/Book.
2. http://www.data-mining-blog.com/data-mining/data-mining-kdd-environment-fayyad-semma-five-sas-spss-crisp-dm/
3. C. Ordonez, “Vertical and Horizontal Percentage Aggregations”, ‘Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’04)’, pp. 866-871, 2004.
4. C. Ordonez, “Statistical Model Computation with UDFs,” IEEE Trans. Knowledge and Data Eng., vol. 22, no. 12, pp. 1752-1765, Dec. 2010.
5. C. Ordonez, “Data Set Preprocessing and Transformation in a Database System,” Intelligent Data Analysis, vol. 15, no. 4, pp. 613-631, 2011.
6. 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, pp. 678- 691, April 2012.
7. Data Representations and Transformations: Illuminating the Path: The R&D Agenda for visual analytics, IEEE, 2005, pp.- 105- 136.