

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 pseudovectors 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.

н 🦷	Agree1						
				Enter Sql	Statement		
sele from whe grou a.na	ct a.name, a. n emp1 a, emp re a.id1=b.id2 up by me, a.city, b.e	city, b.edept,b.ge p2 b edept,b.gender,b.l	nder,b.brcode, prcode	sum(CASE w	hen a.salary> 0) then a.salary El	LSE null END) as salary
	CASE	•	Execute Q	uery	Go to Horizonta	al View	Time : 169.0096
	name	city	edept	gender	brcode	salary	
۶.	Name 1	Bangalore	Aeronatic	Female	189	50000.00	
	Name 1	Bangalore	Aeronatic	Male	127	25000.00	
	Name 1	Bangalore	Civil	Female	135	55000.00	
	Name 1	Bangalore	Computer	Male	118	10000.00	
	Name 1	Bangalore	Computer	Male	137	45000.00	
	Name 1	Bangalore	Electronic	Female	193	20000.00	
	Name 1	Bangalore	IT	Female	116	25000.00	
	Name 1	Bangalore	IT	Female	176	70000.00	
	Name 1	Bangalore	Marine	Female	193	70000.00	
	Name 1	Bangalore	Marine	Male	154	30000.00	
	Name 1	Bangalore	Mechanical	Female	147	75000.00	
	Name 1	Bangalore	Mechanical	Female	155	55000.00	
	Name 1	Bangalore	Mechanical	Male	178	40000.00	
	Name 1	Bangalore	Mechanical	Male	182	10000.00	
	Name 1	Chennei	Civil	Male	181	50000.00	

Figure 3:	Horizontal A	Aggregation an	d Ouerv	Execution
		-888	- CJ	

Aggregate Calar optimizer optimizer Create Mic Heuriserrati View Privery Key (Usiget (b) Total Times MS: 101 0092 Save Result into Datatees Bit Over Result Total Times MS: 101 0092 Save Result into Datatees Optimizer Proventic Cular SOL Code for Honoratel Result Sole Code for Honoratel Result Optimizer Proventic Code for Honoratel Result Sole Code for Honoratel Result Optimizer Proventic Code for Honoratel Result Sole Code for Honoratel Result Sole Code for Honorate Result Into Datatees Sole Code for Honoratel Result Sole Code for Honoratel Result Sole Code for Honorate Result Into Datatees Sole Code for Honoratel Result Sole Code for Honoratel Result Sole Code for Honorate Result Into Datatees Sole Code for Honoratel Result Into Datatees Sole Code for Honoratel Result Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code for Honoratel Result Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code Into Datatees Sole Code Intotees Sole Code Intotees	Aggregatis Optimizer <	Horizontal Column		edept -			Create SC Horizontal View			Erter Ta	ble Name	
Privray Keyr (Visige) Opt Total Time in MS: 101 0002 Stere Result into Obstatement SDL Oxery Result Youtarristic SOL Code for Household Result Sol Code for Household Result Sol Code for Household Result Vig Ammende Code Composite If Sol Code for Household Result Event MS: 101 0002 Event MS	Privary Keyr (Jingan) Opt Cold Time in MS: Base Reade Heta Debatance SDL Colder (France) Marcanetic Cold Colde for Hacconetal Readt SOL Colde for Hacconetal Readt Sol Colde for Hacconetal Readt objective: Americanic Cold Computer Tited Time in MS: Marcanetic Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Cold Colde for Hacconetal Readt Tited Time in MS: Marcanetic Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Cold Colde for Hacconetal Readt Tited Time in MS: Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Cold Colde for Hacconetal Readt Tited Time in MS: Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Colde for Hacconetal Readt Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Colde for Hacconetal Readt Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Colde for Hacconetal Readt Colde for Hacconetal Readt Sol Colde for Hacconetal Readt objective: Sol Colde for Hacconetal Readt Colde for Hacc		Aggregate Column		pregiste Column salary -			Create MC Horizontal View				
SQL Deery Fierd Morantel Read SQL Code for Hexcontal Read obj Auromatic Cold Computer Biolocode Excontex Morine Moderriad Excontex obj Auromatic Cold Code for Hexcontal Read Totopic Biolocode Excontex T Morine Moderriad Excontex obj Auromatic Cold Cold Stoppic Totopic Biolocod Excontex T Morine Moderriad Excontex Hydroxed Stoppic Stoppic Excontex T Morantex Discoppic Discoppic <td< th=""><th>SDL Deery Ruext Mercanetal Ruext SOL Code for Histoprist Ruext object Aurometic Out Out-point Bitschwait T Merina Bechnalt object Aurometic Out Out-point Bitschwait T Merina Bechnalt object Aurometic Out Out-point Bitschwait T Merina Bechnalt Other Area 300000 500000 Exclosed Bitschwait T Merina Bechnalt Other Area 300000 500000 500000 600000 400000 350000 400000 450000 900000 400000 450000 900000 40000 45000 900000 40000 45000 900000 40000 45000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 20000 20000 20000 20000 20000 20000 20000</th><th></th><th>Primery Key /</th><th>Unique</th><th>city</th><th>•</th><th>Total Tm</th><th>e in MS : 161</th><th>0092</th><th>Save R</th><th>lesult into Database</th><th></th></td<>	SDL Deery Ruext Mercanetal Ruext SOL Code for Histoprist Ruext object Aurometic Out Out-point Bitschwait T Merina Bechnalt object Aurometic Out Out-point Bitschwait T Merina Bechnalt object Aurometic Out Out-point Bitschwait T Merina Bechnalt Other Area 300000 500000 Exclosed Bitschwait T Merina Bechnalt Other Area 300000 500000 500000 600000 400000 350000 400000 450000 900000 400000 450000 900000 40000 45000 900000 40000 45000 900000 40000 45000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 40000 20000 20000 20000 20000 20000 20000 20000		Primery Key /	Unique	city	•	Total Tm	e in MS : 161	0092	Save R	lesult into Database	
oby Auronatic Cell Ompount Fit Marine Mestareat Deschard Compount 75000 40000 55000 65000 35000 35000 20000 Operand 39000 40000 55000 60000 20000 55000 40000 Hydroxidad 59000 40000 20000 55000 60000 40000 55000 50000 55000 50000 55000 50000 55000 50000 55000 50000 55000 50000 55000 50000 55000 50000 55000 50000 45000 55000 50000 45000 55000 45000 55000 50000 55000	oby Aurovatic Cell Ouropolar Elserose IT Marine Muchanical Elserose Changleran 750001 650000 650	901	Guery Result	Ha	rurontal Result	SQL Code for Hon	contai Republ					
Surgeoid 725000 6400001 6050001 505000 105000 105000 105000 640000 Cheronal 30000 400000 505000 605000 505000 605000 505000 605000 505000 605000 505000 605000 505000 605000 505000 605000 605000 605000 605000 605000 605000 605000 605000 505000 605000 505000 605000 505000 505000 505000 605000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000 505000<	Surgeoint 725000 640000 650000 6000		city	Aarona	atic Civil	Computer	Electronic	IT	Marine	Mechanical	Bectrool	
Oterval 390000 400000 500000 440000 325000 950000 500000 Hylevelad 200000 550000 500000 505000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 645000 500000 645000 700000 226000 700000 226000 700000 700000 645000 700000 226000 700000 226000 700000 700000 700000 226000 700000 700000 226000 700000 700000 2260000 700000 2260000	Otherwis 590000 440000 610000 440000 325000 950000 450000 Kyshink 65000 590000 500000 450000 <td></td> <td>Dangalore</td> <td>725000</td> <td>640000</td> <td>655000</td> <td>500000</td> <td>635000</td> <td>705000</td> <td>385000</td> <td>620000</td> <td></td>		Dangalore	725000	640000	655000	500000	635000	705000	385000	620000	
Hydroxed 280000 505000 900000 250000 505000 Karher 65500 460000 735000 65000 65000 55000 Martine 775500 45500 45500 45500 45500 45000 Martine 70500 45500 45500 45500 45000 45000 Namber 379500 45500 45500 45000 45000 45000 Namber 379500 45000 45000 45000 95000 55000 Version 53006 60001 65500 45000 95000 95000 Version 530060 605001 15000 45000 95000 50000 Pare 670500 15000 95000 375000 375000 45000 Pare 570500 645000 150500 60000 750000 45000 375000 45000 Pare 570500 645000 150500 60000 750000 450000 320	Hydroxend 20000 59500 290000 290000 940000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 900000 950000		Chennai	390000	490000	590000	610000	4-40000	325000	890000	450000	
Kohn 65000 44000 73500 201000 95500 45500 95000 62000 Mohani 70000 85500 45500 45500 45500 45500 45500 Markai 40000 54500 25500 25500 25500 45500 45500 45500 Namik 57000 82000 49500 24500 35600 45500 35600 45000 35600 45000 35600 45000 35600 45000 35600 35600 20000 35000 <td>Koshni 65000 440000 73500 200000 645000 950000 650000 Modural 705000 655000 455000 455000 455000 455000 Martela 400000 545000 455000 455000 455000 455000 455000 Name 570000 800000 455000 455000 455000 455000 580000 455000 580000 455000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 390000 790000 390000 790000 390000 390000 390000 390000 390000 390000 390000</td> <td></td> <td>Hyderabed</td> <td>280000</td> <td>535000</td> <td>540000</td> <td>270000</td> <td>515000</td> <td>360000</td> <td>360000</td> <td>305000</td> <td></td>	Koshni 65000 440000 73500 200000 645000 950000 650000 Modural 705000 655000 455000 455000 455000 455000 Martela 400000 545000 455000 455000 455000 455000 455000 Name 570000 800000 455000 455000 455000 455000 580000 455000 580000 455000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 580000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 380000 790000 390000 790000 390000 790000 390000 390000 390000 390000 390000 390000 390000		Hyderabed	280000	535000	540000	270000	515000	360000	360000	305000	
Moture 705000 615000 805000 805000 845000 645000 Muntes 40000 56500 62500 645000 645000 Naska 375000 86500 62500 82600 980000 980000 Varias 375000 86500 12500 82600 980000 980000 Varias 375000 81500 125000 126000 980000 980000 Pure 670500 815000 125000 126000 126000 126000 Pure 670500 645000 155000 810000 125000 126000 Pure 570500 645000 1505000 105000 125000 120000	Moture 705000 615000 615000 645000 645000 Martikei 400001 645000 645000 645000 Naskei 370000 645000 950000 950000 950000 Naskei 370000 640000 950000 950000 950000 950000 Outy 330000 610000 615000 950000 325000 950000 Pare 670000 610000 650000 250000 250000 250000 Tixshy 530000 665000 560000 250000 250000 250000		Kochin	635000	490000	735000	200000	505000	485000	595000	520000	
Maraha 410000 545000 102500 770000 470000 980000 410000 Nasik 375000 800000 495000 800000 500000 500000 Cey 320000 800000 495000 425000 500000 500000 Parae 670000 700000 545000 545000 200000 700000 Trichy 520000 645000 560000 700000 200000 200000	Martha 410000 54500 E2500 77000 475000 510000 410000 Namk 57500 80200 49500 402000 55000 50000 Cey 330500 80200 49500 42500 80200 35000 Pare 67000 50000 71000 30200 70000 30000 Trishy 50000 64500 50000 60000 75000 40000 225000		Madurai	705000	615000	495000	385000	490000	480000	455000	645000	
Name 375000 600000 495000 440000 615000 550000 Curry 330000 600000 615000 545000 320000 Pare 670020 330000 610000 545000 375000 670000 Pare 670020 330000 615000 54600 52600 775000 670005 Tachy 530500 645000 500000 500000 720000 465000 502000	Name 375000 600000 495000 240000 610000 550000 Cerry 330000 600000 615000 940000 320000 Pare 670000 300000 615000 940000 22000 320000 Trichy 500000 640000 560000 22000 320000 22000		Mumbai	490000	545000	625000	770000	475000	340000	360000	410000	
Cety 330000 800000 815000 345000 835000 375000 320000 Parke 675001 30000 710000 30000 710000 30000 Trachy 50000 64500 75000 75000 30000 75000 30000	Ceny 330000 603000 51500 342500 802000 372000 320000 Parae 6.70001 300000 710000 300000 670000 Trachy 520000 645000 500000 720000 462000 22000 Trachy 520000 645000 500000 600000 720000 462000 20000		Nasik	375000	680000	495000	245000	4-90000	635000	565000	580000	
Parte Embolio 360000 710000 544000 745000 775000 670000 Techy 505000 645000 500000 500000 700000 205000	Pure 670000 300000 710000 540000 720000 710000 670000 Techy 500000 640000 850000 500000 700000 440000 200000		Oaty	330000	500000	615000	345000	425000	830000	375000	320000	
Truchy 553500 (445000 (853000 5505000 (650000 730000 4460000 2355000	Tnehy 550000 6465000 565000 5600000 750000 465000 205000		Pune	670000	390000	710000	545000	540000	725000	715000	670000	
			Trichy	\$35000	645000	955000	500000	680000	750000	490000	205000	

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.

	d	SPJ		C	ASE	PIVOT	
п	a	DB	Tab-V	DB	Tab-V	DB	Tab-V
1 17	5	30	28	52	45	40.6	23
1K	8	59	44	49	36	50.4	35.4
	10	82	54	69	56	44.8	42
	5	51	48	87	47	53.2	25.4
2K	8	69	56	59	42	63.8	55.6
	10	98	69	84	65	70	56.2
	5	72	68	97	75	68.2	66.2
4K	8	92	82	95	83	88	77
	10	103	96	131	99	103	94.8

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

 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

		S	SPJ	C.	ASE	PIVOT		
n	d	Primary	Secondary	Primary	Secondary	Primary	Secondary	
		Index	Index	Index	Index	Index	Index	
	5	7.91	0.12	7.82	0.112	0.018	0.018	
1K	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	
	5	32.22	0.15	30.34	0.143	0.025	0.025	
2K	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	
	5	127.62	0.17	119.96	0.168	0.046	0.046	
4K	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

		S	SPJ	C.	ASE	PIVOT	
n	d	Primary Index	Secondary Index	Primary Index	Secondary Index	Primary Index	Secondary Index
112	9	7.97	0.121	7.96	0.15	0.031	0.049
IK	11	7.88	0.128	7.93	0.15	0.059	0.043
21	9	32.33	0.257	30.63	0.25	0.056	0.053
21	11	30.49	0.225	38.20	0.26	0.056	0.052
11	9	126.02	0.469	120.66	0.41	0.221	0.074
4 K	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.





🖳 ViewGraph						×
100 ()				1		
100 (sec)						
				SPJ	240	
				CASE	252	
				PIVOT	31	
					Create Graph	
1 (sec)						
	SPJ	CASE	PIVOT			

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 inbuilt 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

- "Data Mining: Concepts and Techniques", Jiawei Han and Micheline Kamber, 1999, ch. 1, Page 3 21 at URL http://db.cs.sfu.ca/Book.
- http://www.data-mining-blog.com/data-mining/data-mining-kdd-environment-fayyad-semma-five-sas-spss-crispdm/
- 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.
- Data Representations and Transformations: Illuminating the Path: The R&D Agenda for visual analytics, IEEE, 2005, pp.- 105- 136.