International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.4, Jul - Aug. 2013 pp-1861-1871 ISSN: 2249-6645

Hortizontal Aggregation in SQL for Data Mining Analysis to Prepare Data Sets

B. Susrutha[1], J. Vamsi Nath[2], T. Bharath Manohar[3], I. Shalini[4]

[1,4]*M. Tech 2ndyr,Dept of CSE,PBRVITS(Affiliated to JNTU Kakinada), Kavali,Nellore.Andhra Pradesh.India.*
[2]*Associate Professor, Dept of CSE, PBRVITS(Affiliated to JNTU Kakinada), Kavali, Nellore, Andhra Pradesh. India.*
[3] *Asst. Professor,Dept of CSE, CMR College of Engineering &Technology,(Affiliated to JNTU Hyderabad) Hyderabad. Andhra Pradesh. India.*

Abstract*: Preparing a data set for analysis is generally the most time consuming task in a data mining project, requiring many complex SQL queries, joining tables and aggregating columns. Existing SQL aggregations have limitations to prepare data sets because they return one column per aggregated group. In general, a significant manual effort is required to build data sets, where a horizontal layout is required. We propose simple, yet powerful, methods to generate SQL code to return aggregated columns in a horizontal tabular layout, returning a set of numbers instead of one number per row. This new class of functions is called horizontal aggregations. Horizontal aggregations build data sets with a horizontal denormalized layout (e.g. point-dimension, observation-variable, instance-feature), which is the standard layout required by most data mining algorithms. We propose three fundamental methods to evaluate horizontal aggregations: CASE: Exploiting the programming CASE construct; SPJ: Based on standard relational algebra operators (SPJ queries); PIVOT: Using the PIVOT operator, which is offered by some DBMSs. Experiments with large tables compare the proposed query evaluation methods. Our CASE method has similar speed to the PIVOT operator and it is much faster than the SPJ method. In general, the CASE and PIVOT methods exhibit linear scalability, where as the SPJ method does not.*

Keywords: *Aggregation, Data Preparation, Pivoting, SQL.*

I. INTRODUCTION

In a relational database, especially with normalized tables, a significant effort is required to prepare a summary data set that can be used as input for a data mining or statistical algorithm. Most algorithms require as input a data set with a horizontal layout, with several Records and one variable or dimension per column. That is the case with models like clustering, classification, regression and PCA; consult. Each research discipline uses different terminology to describe the data set. In data mining the common terms are point-dimension. Statistics literature generally uses observation-variable. Machine learning research uses instance-feature. This article introduces a new class of aggregate functions that can be used to build data sets in a horizontal layout (denormalized with aggregations), automating SQL query writing and extending SQL capabilities. We show evaluating horizontal aggregations is a challenging and interesting problem and we introduced alternative methods and optimizations for their efficient evaluation.

II. MOTIVATION

As mentioned above, building a suitable data set for data mining purposes is a time- consuming task. This task generally requires writing long SQL statements or customizing SQL Code if it is automatically generated by some tool. There are two main ingredients in such SQL code: joins and aggregations; we focus on the second one. The most widely-known aggregation is the sum of a column over groups of rows. Some other aggregations return the average, maximum, minimum or row count over groups of rows. There exist many aggregations functions and operators in SQL. Unfortunately, all these aggregations have limitations to build data sets for data mining purposes.

The main reason is that, in general, data sets that are stored in a relational database (or a data warehouse) come from On-Line Transaction Processing (OLTP) systems where database schemas are highly normalized. But data mining, statistical or machine learning algorithms generally require aggregated data in summarized form. Based on current available functions and clauses in SQL, a significant effort is required to compute aggregations when they are desired in a cross tabular (Horizontal) form, suitable to be used by a data mining algorithm. Such effort is due to the amount and complexity of SQL code that needs to be written, optimized and tested. There are further practical reasons to return aggregation results in a horizontal (cross-tabular) layout. Standard aggregations are hard to interpret when there are many result rows, especially when grouping attributes have high cardinalities. To perform analysis of exported tables into spreadsheets it may be more convenient to have aggregations on the same group in one row (e.g. to produce graphs or to compare data sets with repetitive information).

OLAP tools generate SQL code to transpose results (sometimes called PIVOT). Transposition can be more efficient if there are mechanisms combining aggregation and transposition together. With such limitations in mind, we propose a new class of aggregate functions that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout. Functions belonging to this class are called horizontal aggregations. Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout (somewhat similar to a multidimensional vector), instead of a single value per row. This article explains how to evaluate and optimize horizontal aggregations generating standard SQL code.

III. LITERATURE SURVEY

We have to analysis the DATA MINING Outline Survey:

Data Mining

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

- Automated prediction of trends and behaviors. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.
- Automated discovery of previously unknown patterns. Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

The most commonly used techniques in data mining are:

- Artificial neural networks: Non-linear predictive models that learn through training and resemble biological neural networks in structure.
- Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID) .
- Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.
- Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where k ³ 1). Sometimes called the k-nearest neighbor technique.
- Rule induction: The extraction of useful if-then rules from data based on statistical significance.

An Architecture for Data Mining

To best apply these advanced techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on.

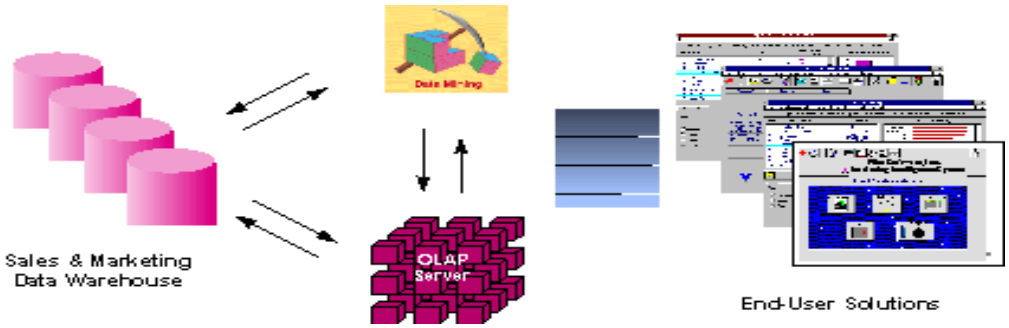


Figure 1: Integrated Data Mining Architecture

The Ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access.

Data Mining Products:

Data mining products are taking the industry by storm. The major database vendors have already taken steps to ensure that their platforms incorporate data mining techniques. Oracle's Data Mining Suite (Darwin) implements classification and regression trees, neural networks, k-nearest neighbors, regression analysis and clustering algorithms. Microsoft's SQL Server also offers data mining functionality through the use of classification trees and clustering algorithms. If you're already working in a statistics environment, you're probably familiar with the data mining algorithm implementations offered by the advanced statistical packages SPSS, SAS, and S-Plus.

Conclusion

Comprehensive data warehouses that integrate operational data with customer, supplier, and market information have resulted in an explosion of information. Competition requires timely and sophisticated analysis on an integrated view of the data. However, there is a growing gap between more powerful storage and retrieval systems and the users' ability to effectively analyze and act on the information they contain. Both relational and OLAP technologies have tremendous capabilities for navigating massive data warehouses, but brute force navigation of data is not enough. A new technological leap is needed to structure and prioritize information for specific end-user problems. The data mining tools can make this leap. Quantifiable business benefits have been proven through the integration of data mining with current information systems, and new products are on the horizon that will bring this integration to an even wider audience of users.

IV. SYSTEM ANALYSIS

Existing System:

An existing to preparing a data set for analysis is generally the most time consuming task in a data mining project, requiring many complex SQL queries, joining tables and aggregating columns. Existing SQL aggregations have limitations to prepare data sets because they return one column per aggregated group.

Disadvantage:

1)Existing SQL aggregations have limitations to prepare data sets.
2)To return one column per aggregated group

Previous Process Flow:

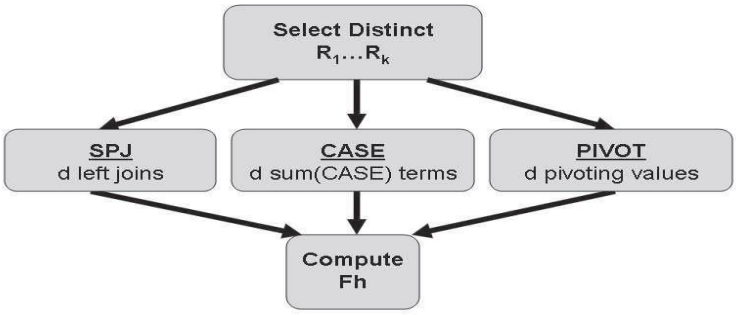


Fig: Previous Process Flow

Proposed System:

Our proposed horizontal aggregations provide several unique features and advantages. First, they represent a template to generate SQL code from a data mining tool. Such SQL code automates writing SQL queries, optimizing them and testing them for correctness.

Advantage:

1) The SQL code reduces manual work in the data preparation phase in a data mining project.
2) The SQL code is automatically generated it is likely to be more efficient than SQL code written by an end user.
3) The data sets can be created in less time.
4) The data set can be created entirely inside the DBMS.

Proposed Process Flow:

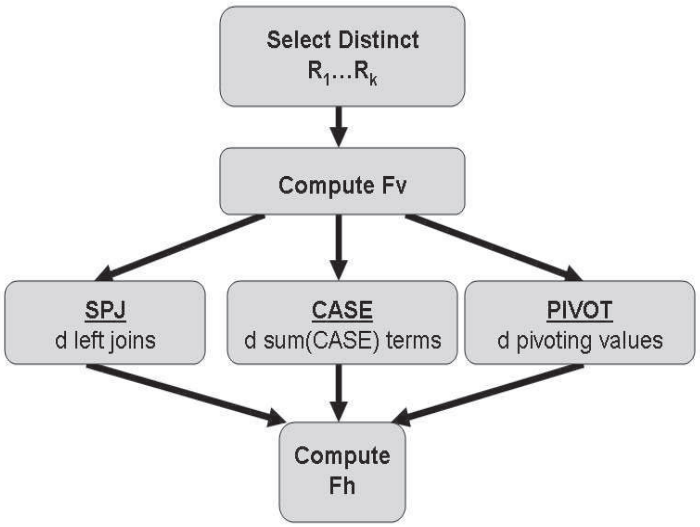


Fig: Proposed Process Flow

Modules Description:

1. Admin Module
2. User Module
3. View Module
4. Download Module

Module 1 : Admin Module

Admin will upload new connection form based on regulations in various states. Admin will be able upload various details regarding user bills like a new connection to a new user, amount paid or payable by user. In case of payment various details regarding payment will be entered and separate username and password will be provided to users in large.

Module 2 : User Module

User will be able to view his bill details on any date may be after a month or after months or years and also he can to view the our bill details in a various ways for instance, The year wise bills, Month wise bills, totally paid to bill in EB. This will reduce the cost of transaction. If user thinks that his password is insecure, he has option to change it. He also can view the registration details and allowed to change or edit and save it.

Module 3 : View Module

Admin has three ways to view the user bill details, the 3 ways are

i) SPJ
ii) PIVOT
iii) CASE

i) SPJ : While using SPJ the viewing and processing time of user bills is reduced.

ii) PIVOT : This is used to draw the user details in a customized table. This table will elaborate us on the various bill details regarding the user on monthly basis.

iii) CASE : using CASE query we can customize the present table and column based on the conditions. This will help us to reduce enormous amount of space used by various user bill details. It can be viewed in two difference ways namely Horizontal and Vertical.

In case of vertical the number of rows will be reduced to such an extent it is needed and column will remain the same on other hand the Horizontal will reduce rows as same as vertical and will also increase the columnar format.

Module 4 : Download Module

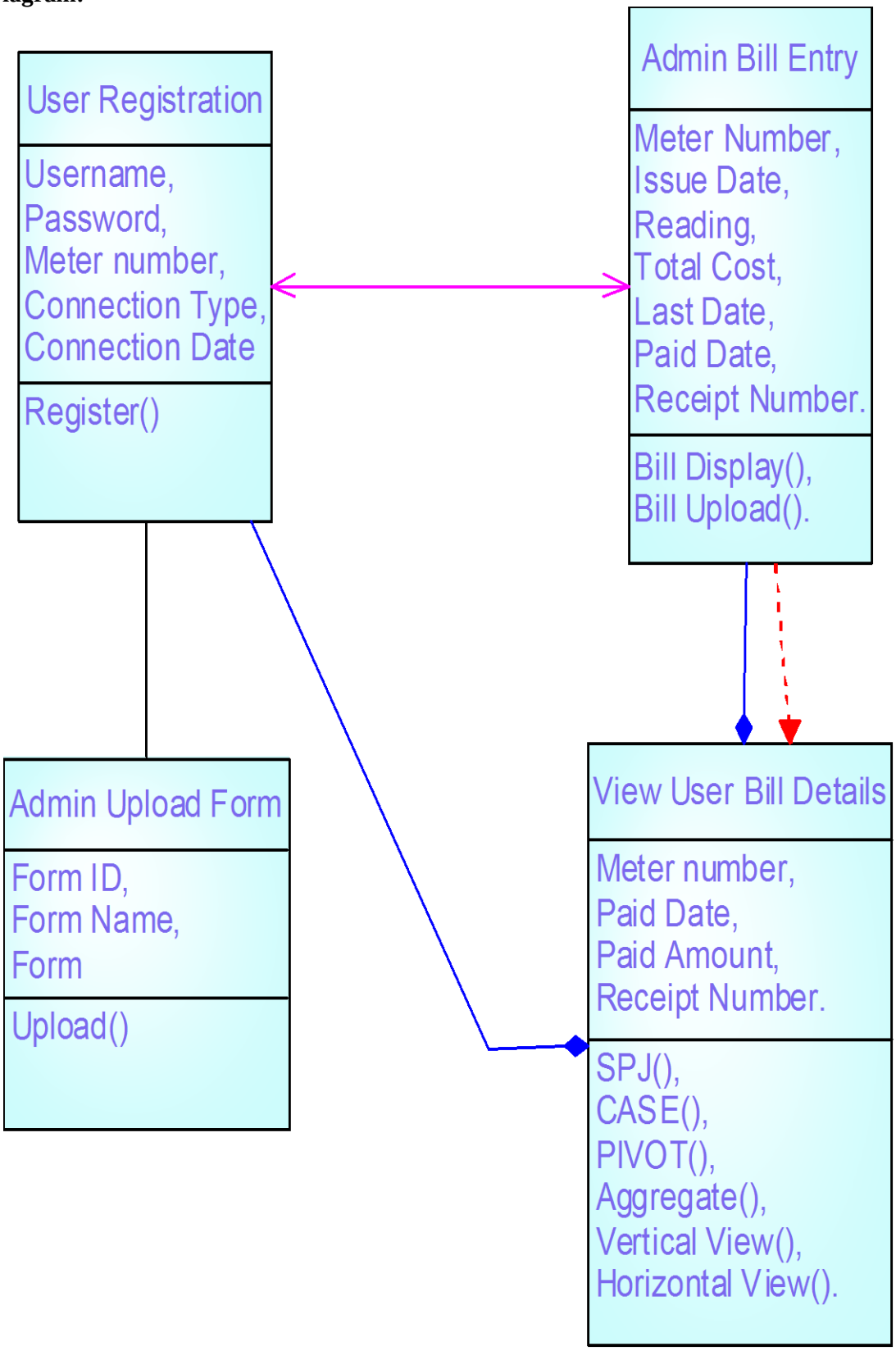
User will be able to download the various details regarding bills. If he/she is a new user, he/she can download the new connection form, subscription details etc. then he/she can download his /her previous bill details in hands so as to ensure it.

V. SYSTEM DESIGN & ARCHITECTURE DIAGRAMS

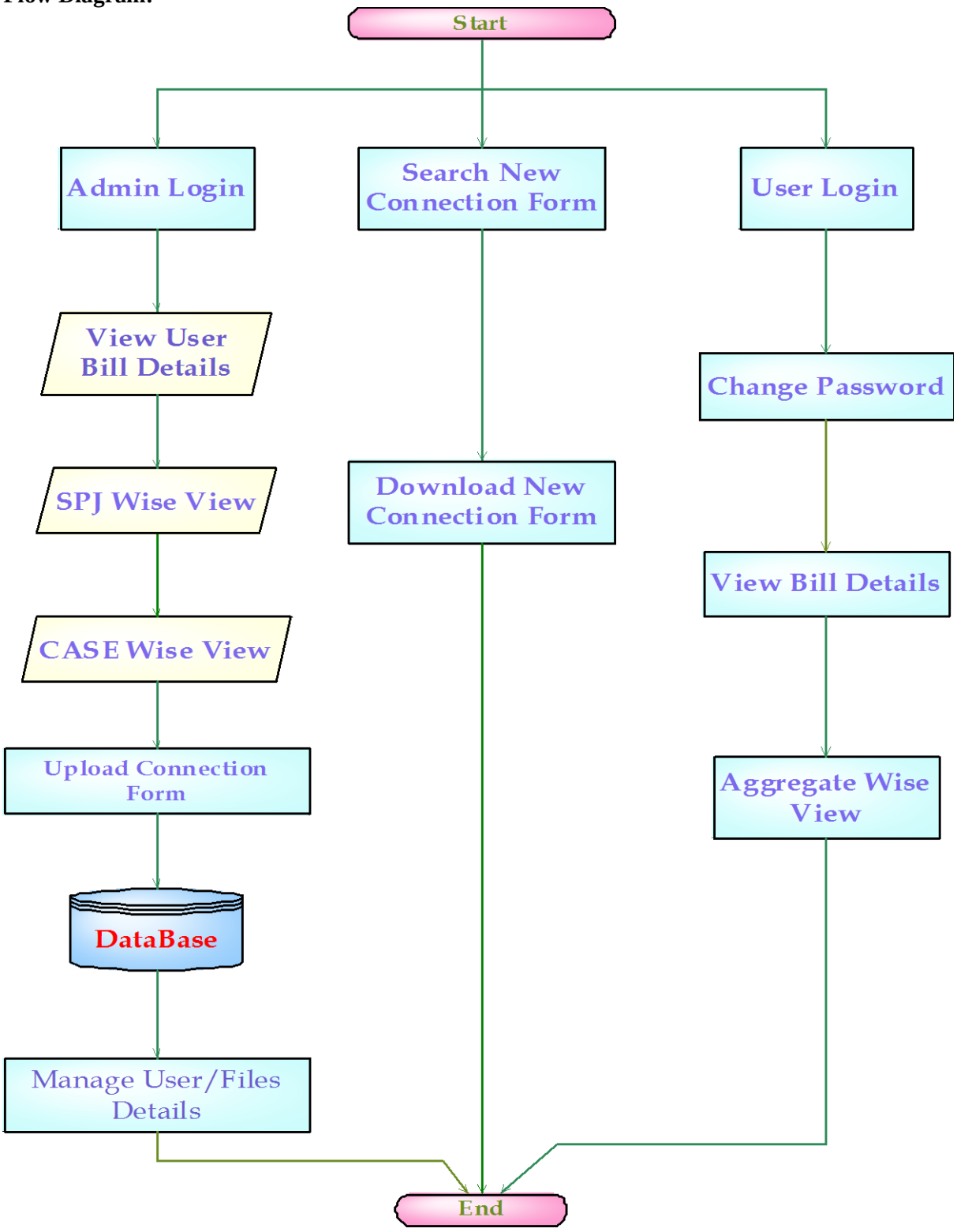
Data Flow Diagram / Use Case Diagram / Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

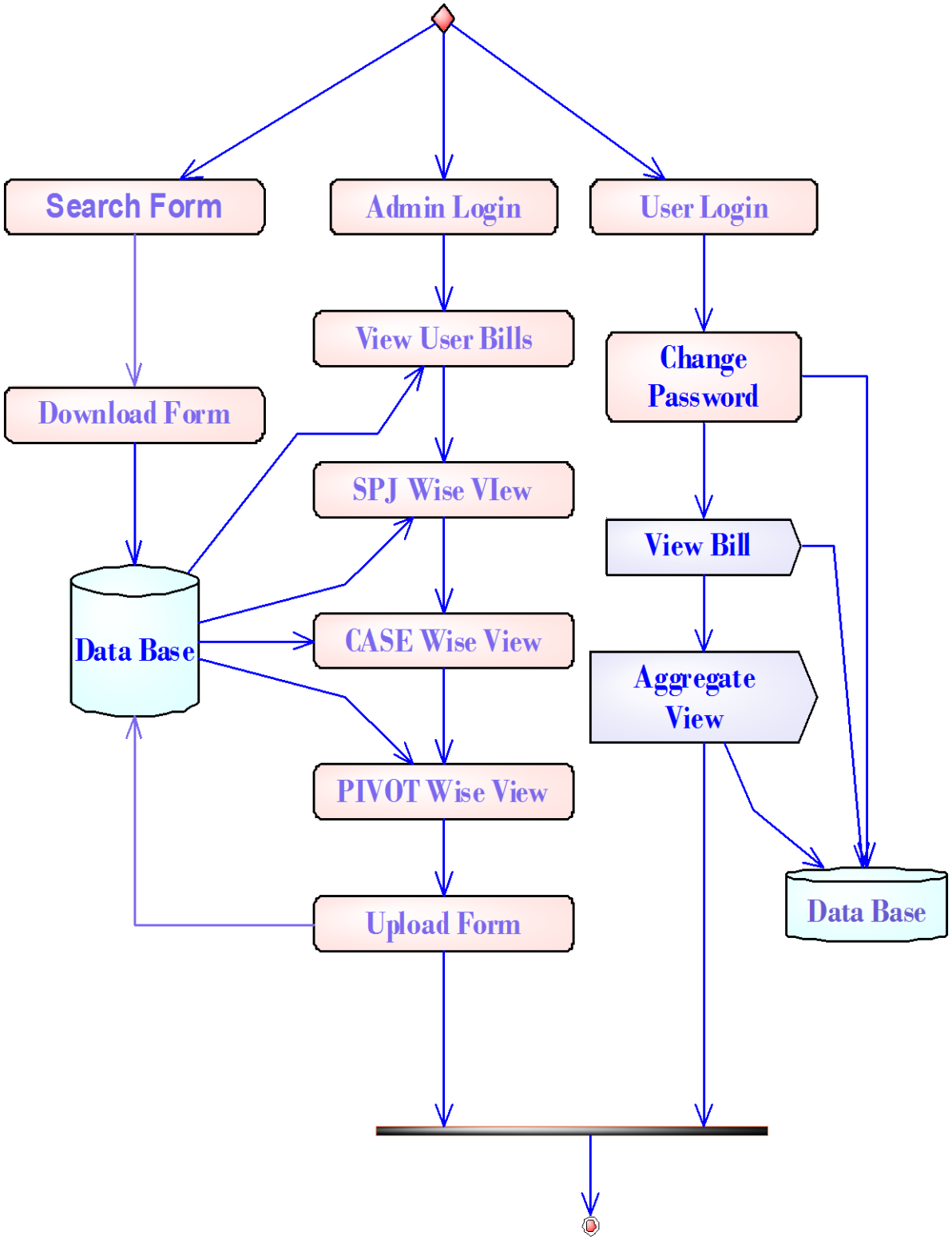
Class Diagram:



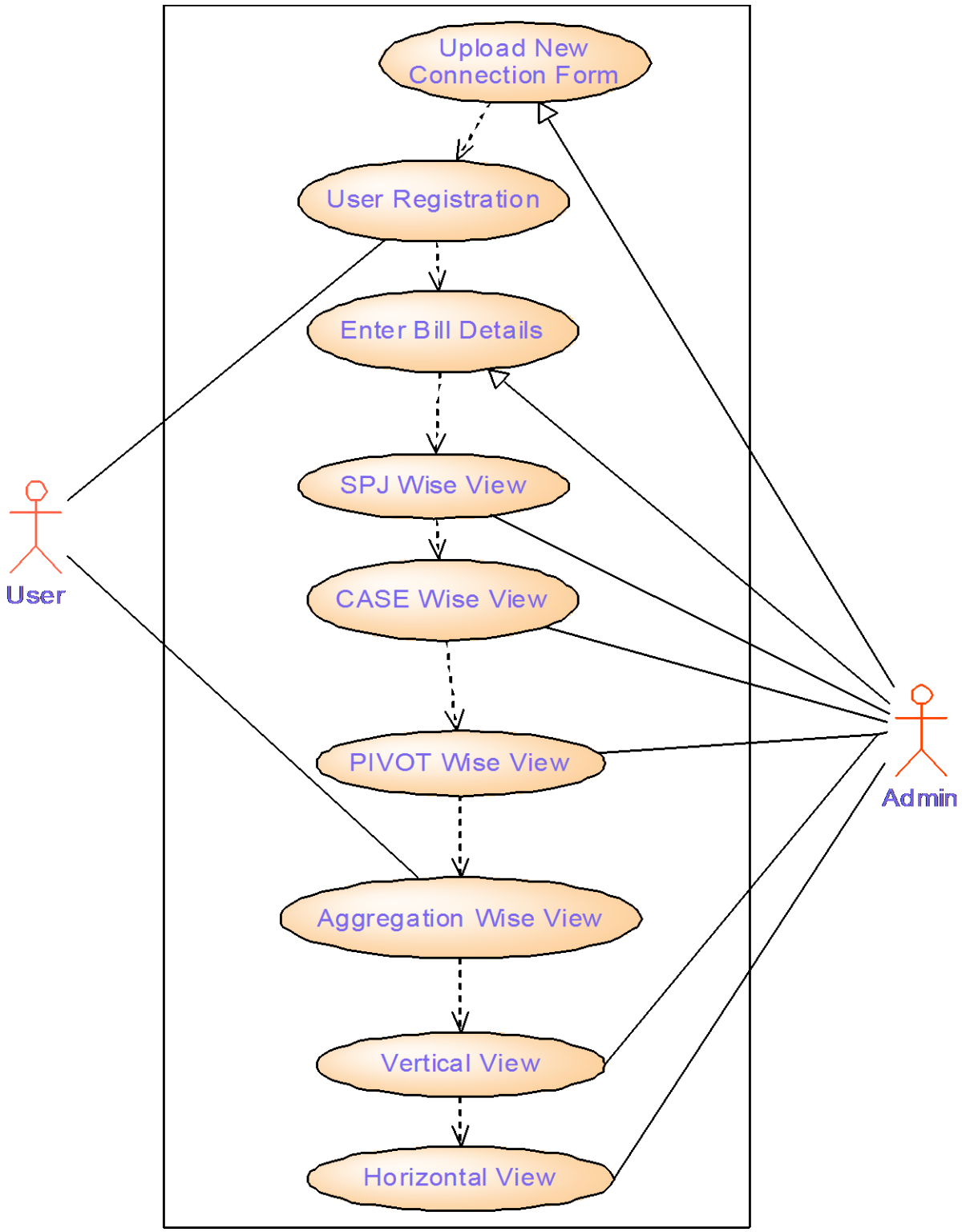
Data Flow Diagram:



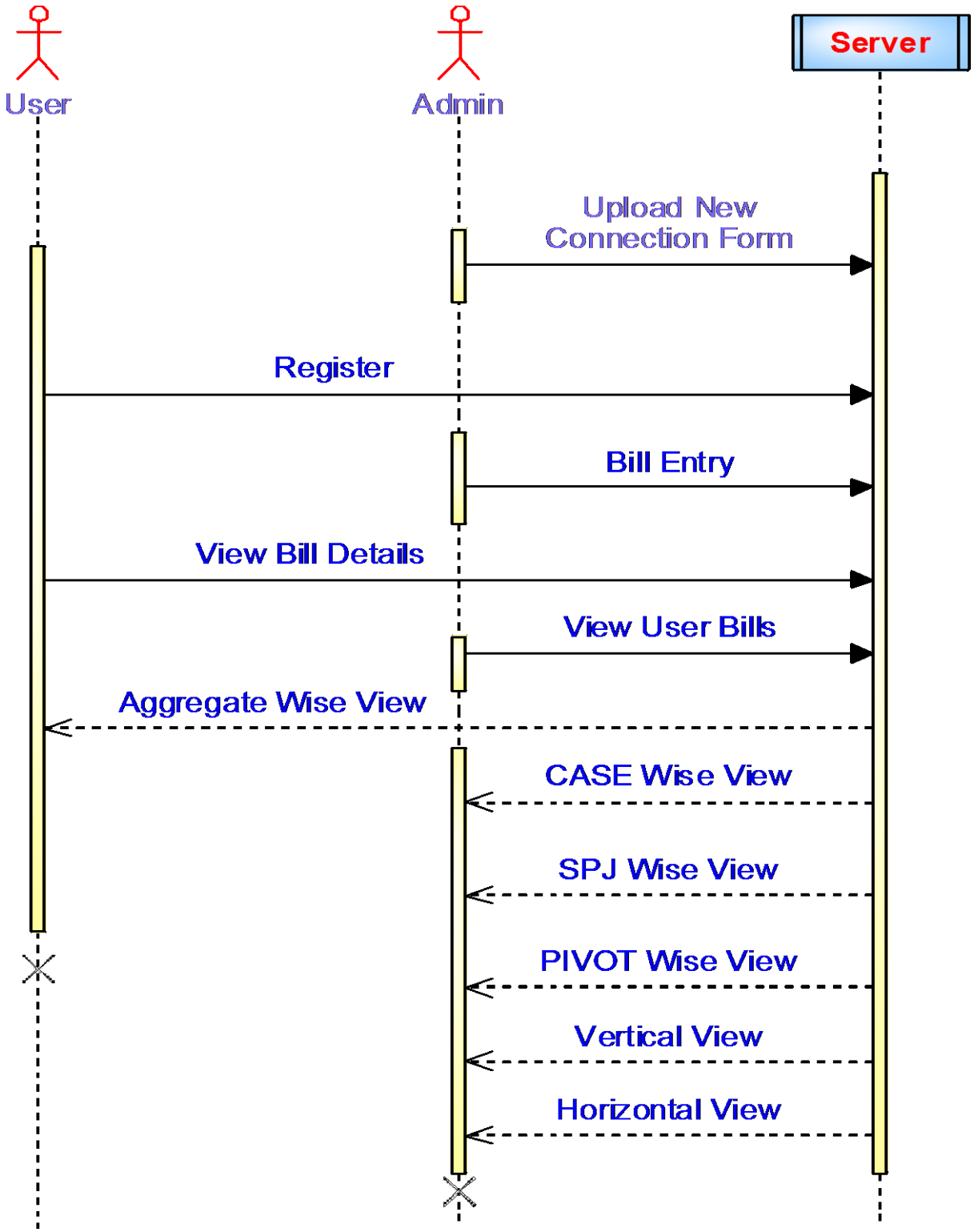
Activity Diagram:



Use case Diagram:



Sequence Diagram:



VI. EXPERIMENT & RESULT

We performed several experiments to test the proposed algorithm and evaluate its performance against different attacks and experienced various results as follows:

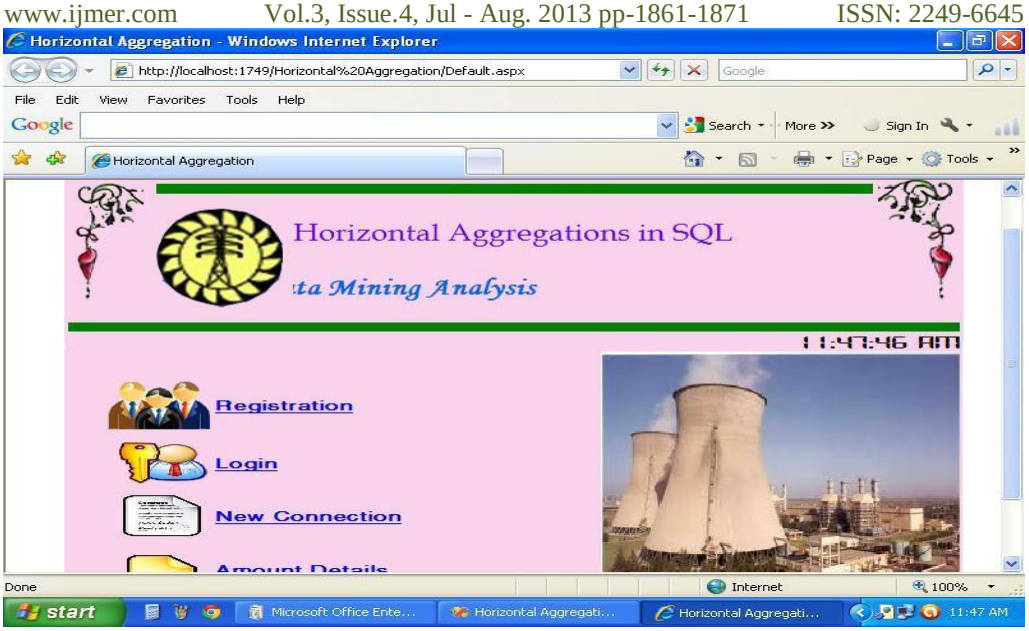


Fig: Horizontal Aggregations in SQL

Fig: Horizontal view results

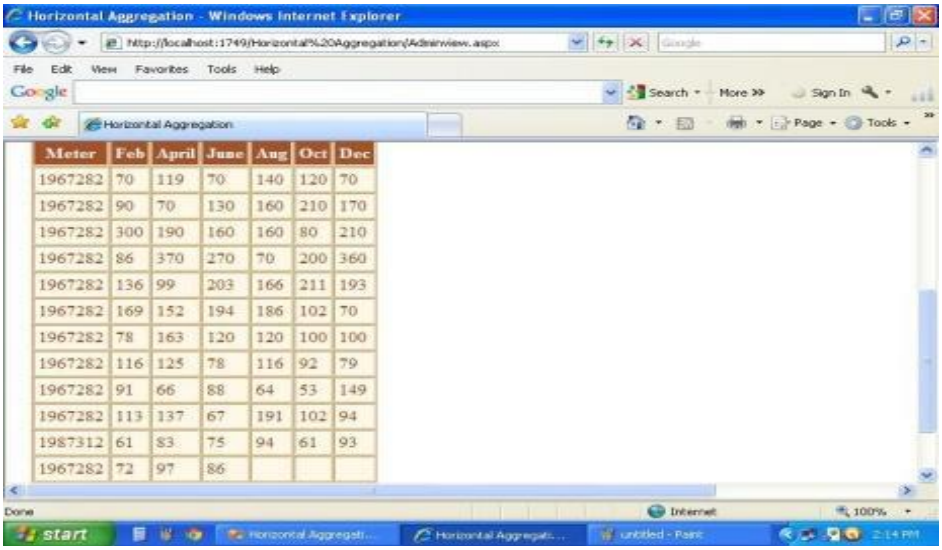


Fig: Results obtained processing the records in Database

VII. CONCLUSION

We introduced a new class of extended aggregate functions, called horizontal aggregations which help preparing data sets for data mining and OLAP cube exploration. Specifically, horizontal aggregations are useful to create data sets with a horizontal layout, as commonly required by data mining algorithms and OLAP cross-tabulation. Basically, a horizontal aggregation returns a set of numbers instead of a single number for each group, resembling a multi-dimensional vector.

We proposed an abstract, but minimal, extension to SQL standard aggregate functions to compute horizontal aggregations which just requires specifying subgrouping columns inside the aggregation function call. From a query optimization perspective, we proposed three query evaluation methods. The first one (SPJ) relies on standard relational operators. The second one (CASE) relies on the SQL CASE construct. The third (PIVOT) uses a built-in operator in a commercial DBMS that is not widely available. The SPJ method is important from a theoretical point of view because it is based on select, project and join (SPJ) queries. The CASE method is our most important contribution. It is in general the most efficient evaluation method and it has wide applicability since it can be programmed combining GROUP-BY and CASE statements. We proved the three methods produce the same result.

We have explained it is not possible to evaluate horizontal aggregations using standard SQL without either joins or ”case” constructs using standard SQL operators. Our proposed horizontal aggregations can be used as a database method to automatically generate efficient SQL queries with three sets of parameters: grouping columns, subgrouping columns and aggregated column. The fact that the output horizontal columns are not available when the query is parsed (when the query plan is explored and chosen) makes its evaluation through standard SQL mechanisms infeasible. Our experiments with large tables show our proposed horizontal aggregations evaluated with the CASE method have similar performance to the built-in PIVOT operator. We believe this is remarkable since our proposal is based on generating SQL code and not on internally modifying the query optimizer. Both CASE and PIVOT evaluation methods are significantly faster than the SPJ method. Precomputing a cube on selected dimensions produced acceleration on all methods.

ACKNOWLEDGMENT

I would like to express my sincere thanks to my Guide and my Co-Authors for their consistence support and valuable suggestions.

REFERENCES

[1] G. Bhargava, P. Goel, and B.R. Iyer. Hypergraph based reordering of outer join queries with complex predicates. In ACM SIGMOD Conference, pages 304–315, 1995.
[2] J.A. Blakeley, V. Rao, I. Kunen, A. Prout, M. Henaire, and C. Kleinerman. .NET database programmability and extensibility in Microsoft SQL Server. In Proc. ACM SIGMOD Conference, pages 1087–1098, 2008.
[3] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman. Non-stop SQL/MX primitives for knowledge discovery. In ACM KDD Conference, pages 425–429, 1999.
[4] E.F. Codd. Extending the database relational model to capture more meaning. ACM TODS, 4(4):397–434, 1979.
[5] C. Cunningham, G. Graefe, and C.A. Galindo-Legaria. PIVOT and UNPIVOT: Optimization and execution strategies in an RDBMS. In Proc. VLDB Conference, pages 998–1009, 2004.
[6] C. Galindo-Legaria and A. Rosenthal. Outer join simplification and reordering for query optimization. ACM TODS, 22(1):43–73, 1997.
[7] H. Garcia-Molina, J.D. Ullman, and J. Widom. Database Systems: The Complete Book. Prentice Hall, 1st edition, 2001.
[8] 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.
[9] 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.
[10] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco, 1st edition, 2001.
[11] 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.
[12] C. Ordonez. Horizontal aggregations for building tabular data sets. In Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop, pages 35–42, 2004.
[13] C. Ordonez. Statistical model computation with UDFs. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22, 2010.
[14] C. Ordonez. Data set preprocessing and transformation in a database system. Intelligent Data Analysis (IDA), 15(4), 2011.
[15] C. Ordonez and S. Pitchaimalai.Bayesian classifiers programmed in SQL. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22(1):139–144, 2010.