Survey on Different Techniques in SQL to Prepare Dataset for Data Mining

Shaila Eksambekar¹, Prof. Suhasini Itkar²

¹PES’s Modern College of Engineering, Shivajinagar, Pune, Maharashtra, India-05
²PES’s Modern College of Engineering, Shivajinagar, Pune, Maharashtra, India-05

Abstract

One of the basic tasks in data mining activity is data pre-processing and preparing dataset. Efficient data analysis can be made easier with datasets having columns in horizontal tabular layout. This paper presents an overview of data pre-processing and dataset preparation techniques using SQL. To prepare dataset if we use SQL aggregations they return one column per aggregated group. This is the limitation of SQL aggregation. In this paper we have proposed need of effective and optimized usage of SQL to build dataset using horizontal aggregations. Also if the result of horizontal aggregation i.e. horizontal layout is integrated with K-means clustering algorithm we can get proper clusters.

Keywords: Data Preprocessing, Dataset, Aggregation, SQL, relational DBMS, SPJ, CASE, PIVOT, K-means.

1. INTRODUCTION

It is a complex task to prepare a summary dataset in a relational database. Such summary dataset is very informative and can be used as input for data mining algorithms. It becomes faster and easier managing large data inside DBMS than outside with alternative tool. This paper presents an overview of different techniques using extended SQL functions and relational DBMS features to prepare datasets, their advantages and challenges that needs attention. After extensive survey and thorough study of related work it is found that using horizontal aggregation functions we can build summary dataset easily. These horizontal aggregation functions prepared in SQL provides opportunity for automation and optimization. Section 2.0 discusses related work (Literature survey). Section 3.0 briefs outcome of this literature survey in terms of advantages of using SQL in dataset preparation task and objective derived out of it. Section 4.0 gives conclusions and direction for future work.

2. LITERATURE SURVEY

This section gives detailed information about related work in data pre-processing using extended SQL techniques and RDBMS features. This literature survey is essential to analyze the importance of the proposed work.

2.1 Using Database system for Pre-processing and Transformation [1]

In this paper disadvantages of manipulating datasets outside the database system are discussed. Many difficulties arise if we manipulate data outside database system. Data integrity is most important issue to be considered under this. Whenever required tables changes or new variables are added, datasets has to be recreated and exported again. If this is not handled properly different users might have access of inconsistent versions of the same dataset. Security is also compromised during this operation as it is not controlled by database system. This paper proposes the idea of transforming datasets and computing models inside an advanced relational database system. They have also emphasized on utilizing many extensive features of the SQL language to prepare dataset. Maximum usage of data management capabilities provided by the DBMS system is one of the objectives of this paper.

Migrating dataset pre-processing and transformation performed by external programs into a database system exploiting the SQL language facilitates efficient computing common sufficient statistics in SQL that benefit several models, which effectively summarize large datasets. As per the paper transforming and storing datasets is much faster inside the database system. In summary, users can exploit the extensive data management capabilities like querying, recovery, security and concurrency control provided by the database system and dataset pre-processing, transformation and analysis are faster inside the database system.

2.2 SQL Extension for Data Mining and Data Streams [2]

H. Wang proposed a new database language ATLaS, which is a powerful database language that enables users to develop SQL applications which are complete data-intensive. He has emphasized on using SQL to write aggregates and table functions in SQL, rather than in procedural languages as in current Object-Relational systems. H. Wang implemented association rule mining using aggregate functions. These aggregate functions are prepared using SQL extensions. In this case, the goal is to efficiently compute item set support. ATLaS’ approach to aggregate definition greatly improves the expressive
power and extensibility of SQL. Although in this paper aggregations are used there is no SQL extension is defined for transposing the results of aggregated data. Transactions are given in a vertical layout.

2.3 SQL Extension for n-dimensional array based computations [3]
Extending SQL with a computational clause which allows us to treat a relation as a multi-dimensional array and specify a set of formulas over it. This not only allows for ease of programming, but also offers the RDBMS an opportunity to perform better optimizations. Joins are avoided by the optimizations mentioned in the paper but partial transpositions for each group of result rows are not optimized. Joins are avoided using the PIVOT and CASE methods. This paper also discusses about single run time access structure which replaces multiple hash or sort structures needed for equivalent joins and UNIONS.

2.4 Vertical and Horizontal Percentage Aggregations [4]
This paper briefs about computing percentage using SQL. It also mentions limitations of existing SQL aggregate functions for percentage computation. To address such limitations, in this paper two SQL aggregate functions are proposed to compute percentages. One percentage per row is calculated using vertical percentage aggregation like standard SQL. Proposed horizontal percentage aggregation returns each set of percentages adding 100 percentages as one row. Existing aggregate functions in OLAP are slower than these proposed aggregations. Horizontal aggregations are more general, have wider applicability in data mining task. To compute percentages these aggregations can be used as a primitive extended operator.

2.5 Integrating Association Rule Mining with Relational database systems [5]
In her paper Sunita Sarawagi has discussed architectural alternatives to integrate mining with database systems. Study has been done about how competitive can a mining computation expressed in SQL be compared to a specialized implementation of the same mining operation. Several potential advantages of a SQL implementation are explained here like use of the database indexing, query processing capabilities, SQL parallelization.

2.6 PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS [6]
This paper proposes use of PIVOT and UNPIVOT SQL operators. When applied on tabular data they exchange rows and columns which enables data transformations useful in data modelling, data analysis and data presentation.

2.7 Horizontal tabular layout generation using Horizontal Aggregation [7]
Ordonez in his paper discusses about how Horizontal Aggregations build datasets with a horizontal denormalized layout. This paper proposes a new class of aggregate functions. Using these functions numeric expressions are aggregated and results can be transposed to produce horizontal layout data set. These functions are called horizontal aggregations. Horizontal aggregations are extended form of traditional SQL aggregations. They return a set of values in a horizontal layout, instead of a single value per row. This paper explains how to prepare horizontal dataset and optimize horizontal aggregations generating standard SQL code.

2.8 Integrating K-Means Clustering with a Relational DBMS Using SQL [8]
This paper proposes integration of K-means clustering algorithm with a relational database. Three SQL implementations of this algorithm are discussed in this paper:
1) Basic version by translating K-means computations into SQL.
2) Optimized version by rewriting queries based on efficient indexing, improved data organization, sufficient statistics and
3) Incremental version- In this version optimized version is used as a building block with automated reseeding and fast convergence.

Horizontal (Summary dataset) layout of the data set enables easier and simpler SQL queries to program clustering algorithm. K-means algorithm implemented in SQL is best suited for partitioning large set of data, obtained from the result of horizontal aggregation, into homogeneous cluster.

3. OUTCOME OF LITERATURE SURVEY
We have major advantages of transforming datasets and computing models in relational database system. Database system through SQL provides powerful querying capabilities. Best data management functionality (maintenance, referential integrity, transaction processing, fault-tolerance, security). Results can be obtained faster and with less programming effort. Using Grouping operators we can get horizontal and vertical aggregations but these are difficult to use when the input dataset is very large. In such case large complex queries are required. ROLL UP operators provide result in only vertical format which is not sufficient for data mining. CUBE operator can be used with large datasets but some details get eliminated. GROUPING COMBINATION operators can be used to get results but they are implemented with complex algorithms so its performance is less. More space is consumed by ATLaS operator than executing with normal SQL. Also it cannot results in horizontal tabular format. Percentage aggregations are similar to normal vertical and horizontal aggregation except that it can compute results only in percentage format. So there may be extra work in the percentage conversion when other computations are required on the dataset. PIVOT and UNPIVOT operators produce output in vertical layout which is not useful in data mining task. UNPIVOT
The main Fig. 1 [7] gives an example of the input table F, a vertical sum() aggregation which we get traditionally stored in F_V, and a summary result in the form of horizontal aggregation stored in F_H.

### SQL Extension with different Syntax:

We have to extend SELECT statement which allows implementation of horizontal aggregations. This extended SQL represents non-standard SQL because columns in the output table are not known when parsing of the query is done. F should be unchanged during evaluation of horizontal aggregation as new values may create new result columns. We extend standard SQL aggregate functions with a transposing BY clause followed by a column list (i.e. T_1,...,T_k), to produce a horizontal set of numbers instead of one number. Suggested syntax is as follows.

```
SELECT G_{i1},...,G_{ij}, H (AggrColumn BY T_{i1},...,T_{ik}) FROM F GROUP BY G_{i1},...,G_{ij};
```

The subgroup columns (i.e. Transposing columns) T_{i1},...,T_{ik} should be a parameter associated to the aggregation. Therefore they should be inside the parenthesis as arguments. Here H() represents some SQL aggregation (e.g. sum(), count(), min(), max(), avg()). H() function must have at least one argument represented by AggrColumn, followed by a column list. Columns G_{i1},...,G_{ij} in the GROUP BY clause (if present) determines result rows. Result columns are determined by all combinations of columns T_{i1},...,T_{ik}, where k=1 is the default. Also, {G_{i1},...,G_{ij}} \cap \{ T_{i1},...,T_{ik}\} =\emptyset. We also have to develop sound and efficient evaluation mechanisms.

Figure 2 [7] and 3 [7] show an overview of the main steps required to prepare dataset in Horizontal Layout. Table 1 & Table 2 give examples of multidimensional dataset which can be created using horizontal aggregations.

---

**Table 1 Example of Horizontal Aggregation [7]**

<table>
<thead>
<tr>
<th>F</th>
<th>F_V</th>
<th>F_H</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>D_1</td>
<td>D_2</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>X</td>
</tr>
</tbody>
</table>

---

**Table 2**

<table>
<thead>
<tr>
<th>Input table F.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Input table F.</td>
</tr>
<tr>
<td>2) List of GROUP BY columns G_{i1},...,G_{ij}</td>
</tr>
<tr>
<td>3) Column to aggregate (AggrColumn)</td>
</tr>
<tr>
<td>4) List of transposing columns T_{i1},...,T_{ik}</td>
</tr>
</tbody>
</table>
3.2 Integrating K-Means Algorithm with Horizontal Aggregation [8]

Integrating data mining algorithms with a relational Database Management System is an important and challenging problem. There are many advantages if we integrate data mining algorithms with a relational DBMS using SQL that is the standard language in relational databases. As we are discussing on horizontal layout, we consider integrating K-Means clustering algorithm which will be taking horizontal dataset as input for further processing. This algorithm partitions a data set into several groups such that points in the same group are similar or close to each other and points across groups are different or far from each other. Advantages of implementing clustering algorithm in SQL are listed:

- As SQL isolates the application programmer from internal mechanisms of the DBMS, application developer does not need to study this mechanism in detail.
- As many data sets are stored in a relational database handling different subsets of data points and dimensions is very flexible, faster, and generally, easier to do inside a DBMS with SQL queries. Otherwise it would be difficult outside with alternative tools.
- Large datasets can be managed easily with DBMS support only.
- DBMS automatically takes care of space management, fault tolerance, secure access, and concurrency control.
- Scalable to very large data sets.

There exist many data mining algorithms that can directly analyse data sets with vertical layout, but to have better I/O pattern algorithms need to be reprogrammed. Also they are efficient only when there many zero values i.e. sparse matrices. As mentioned in paper [8], a horizontal layout of the data set enables easier and simpler SQL queries.

**TABLE 1:** Dataset in Horizontal Layout, Suitable for Data Mining

<table>
<thead>
<tr>
<th>CityID</th>
<th>Scooter_Male</th>
<th>Scooter_Female</th>
<th>Auto_Male</th>
<th>Auto_Female</th>
<th>Vespa_Male</th>
<th>Vespa_Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>30</td>
<td>200</td>
<td>90</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>55</td>
<td>140</td>
<td>85</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>65</td>
<td>220</td>
<td>80</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>77</td>
<td>202</td>
<td>87</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>90</td>
<td>210</td>
<td>100</td>
<td>25</td>
<td>40</td>
</tr>
</tbody>
</table>

**TABLE 2:** A Multidimensional Dataset in Horizontal Layout, Suitable for Data Mining

<table>
<thead>
<tr>
<th>CityID</th>
<th>Sales_Amt</th>
<th>TransactionCount</th>
<th>VehicleCount</th>
<th>Town_Fare</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Sun</td>
<td>Mon</td>
<td>...</td>
<td>Dec</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>...</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>...</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>5</td>
<td>...</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>4</td>
<td>...</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>3</td>
<td>...</td>
<td>10</td>
</tr>
</tbody>
</table>

4. CONCLUSIONS

All the above mentioned papers have their own advantages. Considering all these advantages we can develop a generic framework to automate and optimize data pre-processing tasks using extended horizontal aggregations which would be very useful in data preprocessing task before data mining. Horizontal aggregations will be evaluated using SPJ (Select, Project, Join), CASE construct and built in PIVOT operator. All these evaluation techniques will be compared with respect to time complexity. Left outer joins can be used to evaluate horizontal aggregations and presents opportunities for query optimization. To accelerate computation indexes on common grouping columns, besides indexes on primary keys can be helpful. This study and literature survey can be extended to develop a more generic and formal model of evaluation methods to achieve better results. For cost based query optimization more complete I/O cost models can be developed. We also suggest integration of pre-processed dataset in horizontal layout with clustering model like K-Means for better classification.
References


AUTHOR

Prof. Mrs. Suhasini Itkar is presently working as an Assistant Professor and Head of Department at PES Modern College of Engineering, Pune, India. Her research areas include frequent pattern mining, neural networks and distributed system. She is having 16+ years of experience in teaching. She received B. E. (Computer Science and Engineering) degree in 1997 from Marathwada University, Aurangabad and M. E. (Computer Engineering) degree in 2002 from University of Pune, Pune, India. She joined the Doctoral programme in 2008 at Swami Ramanand Teerth Marathwada University and worked in the area of Data Mining and Distributed System.

Mrs. Shaila Eksambekar is presently working as Senior Technical Officer in C-DAC, Pune. She received B.E. degree in Computer Engineering from Pune University, in 2000. She is now with University of Pune pursuing her M.E. in Computer Engineering.