Efficiency Analysis of Efficient SQL based Clustering Algorithm

Prof.Suresh.L
Prof and Head, Dept. of CSE/ISE
Cambridge Institute of Technology
K.R.Puram, Bangalore -560 036, India.

Dr.Jay.B.Simha
Chief Technology Officer
Abiba Systems
Bangalore, India

Summary
Clustering becomes an indispensable requirement while dealing with immense volume of data. Since Database Management tools does not provide an inbuilt mechanism to cluster datasets of higher magnitude it inevitably requires an external module to perform the same. This external module should be devised specifically to deal with the data extracted from the data source. There exists myriad of techniques for clustering and the conventional approaches deficits like excessive time and computational complexity. Computational complexity becomes a factor of consideration when data is extracted voluminously and the process of clustering and filtration is performed as a subsequent separate operation. The concept proposed is crafted with an object of eliminating the inherent redundancies in the accustomed practice. The proposed solution relies on exploiting the processing power of the Database Management tool by streamlining the SQL used for data extraction. In this paper, we have analyzed our previous work by varying the number of attributes chosen for clustering. And hence a procedure is formulated that can combine data retrieval and clustering to one single operation and leave it to DBMS without letting it to dissipate to the adjoining tiers.

I. Introduction

During the last decades, it has been possible to collect massive data with latest explosion of information and the availability of economical storage. Using this information to gain competitive benefits by discovering formerly unfamiliar patterns in data is the ultimate objective of this massive data gathering. These unfamiliar patterns can direct the decision making. The complicatedness of analyzing data using simply Online Analytical Processing (OLAP) tools, proving the necessity of an automated process in finding out interesting and concealed patterns in data, has been revealed in the recent years. Data mining techniques have ever more been studied [1] [4], especially in their application in real-world databases.

The process of sorting through huge amounts of data and selecting out relevant information is known as Data mining. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases filled in by huge amount of data [2]. Business intelligence organizations and financial analysts generally employ this. To take out information from the large data sets created by modern experimental and observational methods, this is increasingly being used in the sciences. Data mining can be comprehended as a process of extraction of knowledge hidden in extremely large data sets [5] [11].

In an attempt to permit an adequate exploration of data and to deal with the enormous amounts of data that an organization has collected over the years [7], high performance is a key factor for data mining systems. There are two keys to success in data mining. Coming up with an accurate formulation of the problem to be solved is the first key. A concentrated statement generally results in the finest payoff. The second key is using the right data. We may require changing and merging it in considerable ways, subsequent to preferring from the data obtainable to us or possibly buying external data [6]. In order to devour an entity at an instant, data mining algorithms are designed. With relative simplicity, traditional data mining algorithms can be imposed initially. Since it eradicates the requirement for data mining algorithms to do considerably, it enhances scalability subsequently [8].

In the presence of large amounts of data, the data mining algorithms confront the very important issue of scalability. The existing various types of data base technologies contain relational database technologies, hierarchical database technologies, and other types of database technologies as well. A set of inter-related tables, containing rows and columns, have been incorporated by a relational database. It will be a complicated task to administer large data sets without DBMS support. The DBMS is mechanically taking care of space management, fault tolerance, secure access, concurrency control, etc., for the majority part. The time to export it to a workstation can be considerably important [18], even though it is probable to competently cluster an extremely large data set outside a relational database.

In data mining, Clustering is one of the most significant tasks [3]. Clustering divides a database into different groups. A number of algorithms appropriate to cluster extremely large databases are available, and a few of those focus on high-dimensional data [13], [14], [19], [20], [21] and [22]. Clustering is a division of data into groups of comparable objects. Fewer clusters symbolizing data

Manuscript received November 5, 2008
Manuscript revised November 20, 2008

essentially misplaces certain fine details, however attains simplification. In data mining applications for instance scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, medical diagnostics, computational biology, and many others, clustering plays a crucial role. Data mining enhances clustering of the complications of extremely large datasets with many attributes of dissimilar types. This imposes distinctive computational necessities on applicable clustering algorithms. Various clustering algorithms, meeting these necessities, have been newly emerged and were effectively applied to real-life data mining problems [9].

In the data mining literature, a lot of competent clustering algorithms exist. A few of them are Hierarchical Methods, Partitioning Methods, Probabilistic Clustering, K-means Methods [21], [23], Density-Based Algorithms, Density-Based Connectivity Clustering, Density Functions Clustering, Grid-Based Methods, Scalable Clustering Algorithms, Algorithms for High Dimensional Data [13], [14], [19], [20], [21] and [22]. etc. The disk access has been lessened by the majority of the approaches and doing most of the work in main memory. Regrettably, executing a real DBMS is extremely tough in many of those algorithms where the programmer requires to be concerned about storage management, concurrent access, memory leaks, fault tolerance, security and so on [16].

In this paper, we have analyzed our previous work by varying the number of attributes chosen for clustering [27] with relational databases using SQL, the standard interactive and programming language for querying and modifying data and managing data in relational databases has been principally focused. In this, the number of attributes chosen for clustering is three whereas it was two in the previous work. This command language permits the recovery, inclusion, updating, and removal of data, and too executing management and administrative functions.

1.1 Structured Query Language (SQL)

Structured Query Language (SQL) is an ANSI standard for accessing and manipulating relational database content. It is extensively used in industry and is supported by major database management systems (DBMS). For textual and numeric data, DBMS are first and foremost used. These textual and numeric data have been divided into records and fields [15]. SQL is available in every relational DBMS. To turn into fairly comprehensive and complex query language where the features are automatically managed or it can be adjusted by the database application programmer, SQL has been rising over the years. Furthermore, to interact with a relational DBMS, it is the distinctive approach. Because, this approach recuperates the speed, storage management, concurrent accesses, etc. [16]. With the facility to recover records independently from the database, carry out all computation in main memory, and write any required changes back to the databases [17], the basic idea employs the DBMS as a simple storage.

The application programmer has been secluded by SQL from internal mechanisms of the DBMS. In a relational database and dissimilar subsets of data points, numerous data sets are stored. To do inside a DBMS with SQL query, dimensions are more flexible, faster and easier than outside with alternative tools [16]. Application downtime, increased scalability and performance, and flexible security controls have been condensed by SQL. To deploy, handle, and optimize enterprise data and analytical applications, SQL creates it simpler and easier. Many benefits in terms of speed, scheduling capabilities, data analysis, security, scalability, and reliability have been presented by the implementation of SQL in data mining. To execute complex mathematical operations, SQL has a few restrictions. To manipulate matrices, SQL never provide arrays and functions. As a result, if feasible to express in SQL, many computations can turn into unwieldy. SQL queries incur higher overheads than directly accessing the file system with an elevated programming language like Java [16].

The paper is organized as follows. In section 2, the related works of this paper is given. In Section 3, the proposed SQL based clustering algorithm is discussed in detail. In section 4, the experimental results are presented and conclusions are summed up in Section 5.

2. Related Works

Beibei Zou, et al., have presented an approach to the integration of learning algorithms with relational databases and built an extension of the well-known Weka data mining library, WekaDB, which allows the data used by the learning algorithms to reside on secondary storage. This change is transparent to the developers of machine learning algorithms [17].

A data clustering method named BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), have presented by Tian Zhang, et al., and demonstrates that it is especially suitable for very large databases. It makes a large clustering problem tractable by concentrating on densely occupied portions, and using a compact summary. It utilizes measurements that capture the natural closeness of data. BIRCH can work with any given amount of memory, and the I/O complexity is a little more than one scan of data [22].

The new API for data mining has proposed by Microsoft as extensions to OLE DB standard. Amir Netz, et al., have illustrated the basic notions that motivated the API's design and describe the key components of an OLE DB for Data Mining provider. It also includes examples of the usage and treats the problems of data representation and integration with the SQL framework. Also this new API will go a long way in enabling deployment of data mining in enterprise data-warehouses [8].

Carlos Ordonez introduced two implementations of K-means in SQL. The proposed implementations allow clustering large data sets in a relational DBMS eliminating the need to export or access data outside the DBMS. Only standard SQL was used; no special data mining extensions for SQL were needed. To cluster very large data sets in a single scan using SQL combining the ideas proposed here with User Defined Functions, OLAP extensions, and more efficient indexing [12].

Carlos Ordonez introduced three SQL implementations of the popular K-means clustering algorithm to cluster large data sets and integrate it with a relational DBMS: 1) a straightforward translation of K-means computations into SQL, 2) an optimized version based on improved data organization, efficient indexing, sufficient statistics, and rewritten queries, and 3) an incremental version that uses the optimized version as a building block with fast convergence and automated reseeding [16].

Clustering was originally introduced as a fast, scalable solution for large multidimensional databases with numeric values. Here, Boriana L. Milenova, et al., extended O-Cluster to domains with nominal and mixed values. O-Cluster uses a top-down partitioning strategy based on orthogonal projections to identify areas of high density in the input data space. The algorithm employs an active sampling mechanism and requires at most a single scan through the data. We demonstrate the high quality of the obtained clustering solutions, their explanatory power, and O-Cluster's good scalability [24].

Sudipto Guha, et al., proposed a clustering method called CURE. CURE utilizes multiple representative points for each cluster that are generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. This enables CURE to adjust well to the geometry of clusters having non-spherical shapes and wide variances in size. To handle large databases, CURE employs a combination of random sampling and partitioning that allows it to handle large data sets efficiently [25].

Chris Ding and Xiao feng proved that the continuous solutions of the discrete K-means clustering membership indicators are the data projections on the principal directions (principal eigenvectors of the covariance matrix). New lower bounds for K-means objective function are derived, which relate directly to the Eigen values of the covariance matrix [26].

3. SQL Based Clustering Algorithm

The efficient database management systems have been very important assets for management of a large amount of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed.

There are many existing clustering algorithms which work more efficiently to cluster the large datasets [13], [14], [19], [20], [21] and [22]. One of the important among them is k-means clustering algorithm. If there are huge amount of records in the dataset, the records are to be clustered very efficiently according to their related fields. Cluster a large datasets using k-means algorithm with relational database seems to be more iterative and requires more time. But using SQL the massive datasets can be clustered very effectively.

So we have proposed an algorithm namely SQL based clustering algorithm which clearly explains about clustering the large data set using SQL with minimum time duration. The goal of clustering is to find groups that are very different from each other, and whose members are very similar to each other. SQL queries have been used to retrieve data in the form of contingency table, which is further used to extract patterns in the data [9]. The number of fields present in the large datasets can be clustered by using SQL queries. The initial set of the related groups are efficiently clustered by using the SQL with minimum amount of time. Then, with the help of our proposed framework, the datasets can be clustered very effectively.

The enormous amount of datasets has been clustered very efficiently by using this SQL based clustering algorithm. The efficiency of this algorithm is to attain the peak improvement by means of reducing the iteration time and it is the basic limitation in other clustering techniques. Also, it is more efficient only if the key fields chosen for clustering have obtained significant redundant data, the SQL algorithm will perform better than other conventional algorithms.

The pseudo code and its explanation are given below,

Assumptions

T_d	→ Table contains input data sets.
T_g	→ Table contains grouped data sets.
T_{gd}	→ Table contains index and Euclidean distance of grouped data sets.
T_c	→ Table contains clustered data sets.

NF → Number of fields

The required number of fields (NF) mainly depends on the amount of data which is given by the user. As the amount of data increases the number of usage of fields will also increases.

In the first stage of the clustering the redundant data in the different fields have been merged by using a query q which is given by the equation as

$$q = \sum_{i=0}^{NF-1} f_1 * pow(10, i)$$

Where f_1 is the data field.

The SQL script for inserting the merged fields is given by

INSERT INTO T_g

SELECT $+q+$

from T_d

GROUP BY $+ g$;

The merged fields and their ids are grouped and finally it will be inserted into the table T_g . Hence only the id values and their corresponding field values will reside in the table T_g .

E_d → Euclidean Distance

INSERT INTO T_{gd}

$(Index1$, Index2, value1, value2, Euclidean Distance)

SELECT $g_1.id, g_2.id, g_1.f_{1,} g_2.f_1, Ed(g_2.f_1 - g_1.f_1)$

FROM T_g

WHERE $(([g_1].[id]+1) = [g_2].[id])$

ORDER BY Ed $(g_2.f_1 - g_1.f_1)$

The Euclidean distance is calculated by the difference between two consecutive values of grouped fields. The values will be sorted in ascending order and it will be inserted into the table T_{gd} which contains indices, values and corresponding Euclidean distances.

Then the SQL script to get the values from the table T_{gd} is given by

SELECT Index1, value1, value2, Euclidean Distance from T_{gd}

Assumptions

I_{DS}	→ Vector having the index of the grouped data set.
S_D	→ Vector having grouped data set.
I_C	→ Index of cluster
K	→ Number of clusters.
De	→ Euclidean distance of grouped data set.

Set rs as Record Set;

The values of the table T_{gd} is taken as a record set for further programming.

```
$d \leftarrow 0$;
$g \leftarrow 0$;
$a \leftarrow 0$;

while $(rs != empty)$ do
    $I_{DS}[0] \leftarrow rs.getInt(1)$;
    $I_{DS}[1] \leftarrow I_{DS}[0]+1$;
    $S_D[0] \leftarrow rs.getInt(2)$;
    $S_D[1] \leftarrow rs.getInt(3)$;
    $De \leftarrow$ rs.getInt(4);
    if g < K then
        if De == d then
            if (a!= $I_{DS}[0]$) then
                $g \leftarrow g+1$;
            end if
        else
            $g \leftarrow g+1$;
        end if
        for each element i in $S_D$
            insert into $T_C$ values (g, $S_D[i]$) ;
            $I_C[g] \leftarrow S_D[1]$;
        end for
    else
        $s \leftarrow \Phi$; where $\Phi$ is a greater value
        $v \leftarrow 0$;
        $ind \leftarrow 0$;
        for each element i in $S_D$
            for each element j in $I_C$
                $v \leftarrow abs(I_C[i] - S_D[j])$;
                if $s > v$ then
                    $s \leftarrow v$;
                    $ind \leftarrow j$;
                end if
            end for
```

```
                  $insert$ into $T_C(g, S_D[i])$
              $I_C[ind] \leftarrow S_D[i];$
              $end$ for
        $end$ if
        $d \leftarrow De;$
        $a \leftarrow I_{DS}[1];$
$end$ while
```

The results of the proposed algorithm i.e. clustered data are stored in the concluding table T_c.

The indexes of the grouped dataset are taken from the record set. Then the values of the different Euclidean distance are assigned in different clusters. This operation will be performed up to K number of clusters. The remaining values are assigned to the cluster which is having minimum difference between the Euclidean distances of the two values. This will be continued up to all the values in the record set. Eventually, the clustered data are inserted into the concluding table T_c.

4. Experimental Results

This section contains an extensive experimental evaluation. We have implemented the proposed SQL based clustering algorithm in Java. In the proposed algorithm, the redundant data in the selected fields have been grouped initially with minimum amount of time. The results have been analyzed by comparing the k-means algorithm with the proposed algorithm by varying our previous work. In the previous work, the results have been analyzed against the k-means algorithm with the number of fields is two. In this work, we have enhanced the previous work by increasing the number of fields to three and analyzed the results of the proposed algorithm.

The main limitation of k-means implementations is that the datasets needs to be scanned several times. So this algorithm requires more time for clustering than our proposed solution. In this, we have compared the time to export data sets by the K-means iteration. We emphasize that the iteration operation is performed few times for one data set, whereas K-means iterations are repeated many times. But our algorithm performs better than k-means and also the results are compared with the k-means algorithm by means of reducing the computational complexity and time consumption. We have compared both the SQL Query clustering and K-means clustering results with large throughputs and those experimental results are listed in charts as follows. Figure 1 shows the time analysis of our proposed clustering algorithm and Figure 2 shows the time analysis of K-means clustering algorithm. In the chart, the x-axis is represented by the quantity of datasets and y-axis is represented by the time needed for clustering those datasets. From the graph, we conclude that our proposed algorithm outperforms k-means algorithm in execution time.

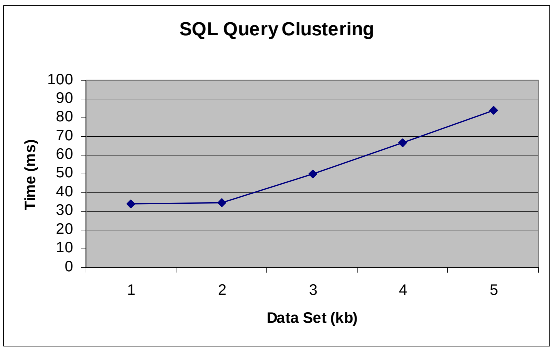

Fig.1 Time analysis of SQL based clustering algorithm

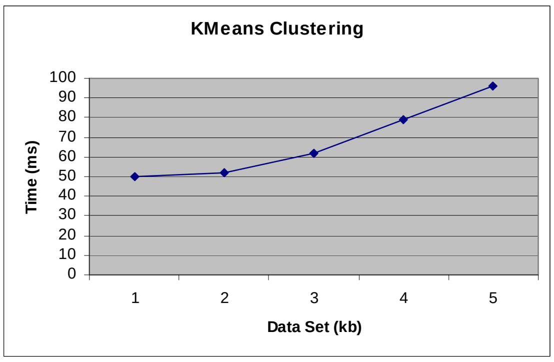

Fig. 2 Time analysis of K-Means clustering algorithm

5. Conclusion

Clustering large datasets has some important issues like excessive time, computational complexity and so on. The existing works have not specified the effectiveness of database management systems using SQL in clustering algorithms. So we have proposed an efficient clustering algorithm in this framework by using SQL. The SQL algorithm is used to cluster a large datasets very efficiently than the other existing clustering techniques, only if the key fields chosen for clustering have obtained significant redundant data. This algorithm gives the better results when compared with the k-means algorithm by the way of reducing the computational complexity and time consumption.

References

[1] J. P. Bigus., "Data Mining with Neural Networks", McGraw-Hill, 1996.

[2] Baloglu, A., and Abdel-Badeeh M, "Intelligent Web- Based System: A Proposed Model for E-Lab Services", International Journal of Soft Computing Applications, No.2, pp.5-14, 2007.

[3] Fayyad U, Piatetsky G., Smyth P., Uthurusay R., "Advances in Knowledge Discovery and Data Mining", AAAI /MIT press, Cambridge,1996.

[4] T. M. Mitchell., "Machine Learning", McGraw-Hill, 1997.

[5] Ralf Rantzau and Holger Schwarz, "A Multi-Tier Architecture for High-Performance Data Mining", Institute of Parallel and Distributed High-Performance Systems (IPVR), pp. 20-22, 1999.

[6] Herbert A. Edelstein, "Introduction to Data Mining and Knowledge Discovery", 3rd edition, Published in two crows corporation, 1999.

[7] J. Zytkow and S. Gupta , "Mining Medical Data Using SQL Queries and Contingency Tables", Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases , 2001.

[8] Netz, A. et al, "Integrating Data Mining with SQL Databases: OLE DB for Data Mining", Proceedings of the 17th International Conference on Data Engineering, pp. 379-387, 2001.

[9] Pavel Berkhin, "Survey of Clustering Data Mining Techniques- technical report", Accrue software, pp. 1092-1460, 2002.

[10] Lorinda Visnik, "Clustering Techniques", A technical White paper, pp. 2-8, 2002.

[11] Ramesh Natarajan, Radu Sion, Chid Apte, and Inderpal S. Narang, "A Grid-based Approach for Enterprise-Scale Data Mining", Future Generation Computer Systems, Vol. 23, pp. 48-54, 2004.

[12] Carlos Ordonez, "programming the k-means clustering in SQL", Proceedings of the 10th ACM / SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 823- 829, 2004.

[13] Yihong Dong, Yueting Zhuang, "Fuzzy hierarchical clustering algorithm facing large databases", Fifth World Conference on Intelligent Control and Automation (WCICA), Vol. 5, pp. 4282 - 4286, 2004.

[14] Zhijie Xu, Laisheng Wang, Jiancheng Luo, Jianqin Zhang, "A modified clustering algorithm for data mining", Proceedings of the IEEE International conference on Geoscience and Remote Sensing Symposium, ,Vol. 2 , pp. 4 , 2005.

[15] Wolfram, Dietmar, "Applications of SQL for Informetric Data Processing", Proceedings of the 33rd conference of the Canadian Association for Information Science, 2005.

[16] Carlos Ordonez, "Integrating K-Means Clustering with a Relational DBMS Using SQL", IEEE transactions on knowledge and data engineering, Vol. 18, No.2, pp. 188-201, 2006.

[17] Zou, B., Ma, X., Kemme, B., Newton, G , Precup, "Data mining using Relational Database Management Systems", Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Vol. 3918, pp. 657-667, 2006.

[18] Carlos Ordonez, "k-means clustering using structured query language statements and sufficient statistics", NCR Corporation, 07359913, Cl. 707-102, 2008.

[19] R.J. Gil-Garcia; J.M. Badia-Contelles; A. Pons-Porrata, "A General Framework for Agglomerative Hierarchical Clustering Algorithms", Pattern Recognition, 2006. ICPR 2006. 18th International Conference on Volume 2, Issue, 2006 Page(s):569 – 572

[20] Liping Jing; Ng, M.K.; Huang, J.Z. ,"An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data", Knowledge and Data Engineering, IEEE Transactions on Volume 19, Issue 8, Aug. 2007 Page(s):1026 – 1041.

[21] C. Ordonez, "Clustering Binary Data Streams with K-Means," Proc. ACM Data Mining; ACM Data Mining and Knowledge Discovery Workshop, pp. 35-41, 2003

[22] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An efficient data clustering method for very large databases". In ACM SIGMOD Conference, pages 103{114, 1996.

[23] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. Applied Statistics, 28(1):100--108, 1979.

[24] B. L. Milenova and M. M. Campos, "Clustering large databases with numeric and nominal values using orthogonal projections", white paper, 2003.

[25] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in ACM SIGMOD International Conference on Management of Data, pp. 73-84, 1998.

[26] Chris Ding and Xiao feng, "Principal Component Analysis and Effective K-means Clustering",In Proceedings of SIAM International Conference on Data Mining, 2004.

[27] L.Suresh and Jay.B.Simha, "Novel and Efficient Clustering Algorithm Using Structured Query Language," International Conference on Computing Communication and Networking (ICCCN), December 18 - 20, 2008. [Accepted for Publication].

Suresh L is working as a Professor & Head, Department of Computer science/Information Science and Engineering at Cambridge Institute of Technology, Bangalore, INDIA. He has received his B.E., M.S. in Computer Science & Engineering and pursuing his Ph.D at Dr.M.G.R.University, Chennai, Tamilnadu, INDIA. His Research area is Data Mining.

Dr. Jay B.Simha is working as a Chief Technology Officer at Abiba Systems, Bangalore, INDIA. He has received Ph. D. degree in Computer Science from Bangalore University in 2003. His research areas are Data Mining, Fuzzy Logic and Artificial Intelligence.