Automatic Database Clustering: Issues and Algorithms

Sakshi Kumar *
CSE Department, JBKP, MDU

Mahesh Singh
CSE Department, AITM, MDU

Sunil Sharma
CSE Department, JBKP, MDU

ABSTRACT**—** Clustering is the process of grouping of data, where the grouping is established by finding similarities between data based on their characteristics. Such groups are termed as Clusters. Clustering is an unsupervised learning problem that group objects based upon distance or similarity. While a lot of work has been published on clustering of data on storage medium, little has been done about automating this process. There should be an automatic and dynamic database clustering technique that will dynamically re-cluster a database with little intervention of a database administrator (DBA) and maintain an acceptable query response time at all times. A good physical clustering of data on disk is essential to reducing the number of disk I/Os in response to a query whether clustering is implemented by itself or coupled with indexing, parallelism, or buffering. In this paper we describe the issues faced when designing an automatic and dynamic database clustering technique for relational databases.. A comparative study of clustering algorithms across two different data items is performed here. The performance of the various clustering algorithms is compared based on the time taken to form the estimated clusters. The experimental results of various clustering algorithms to form clusters are depicted as a graph.

KEYWORDS**—** Cluster, Cluster Analyzer, Database Clustering, Nodes,

I. Introduction

Databases, especially data warehouses and temporal databases, can become quite large. The usefulness and usability of these databases highly depend on how quickly data can be retrieved. Consequently, data has to be organized in such a way that it can be retrieved efficiently. One big concern when using such databases is the number of I/Os required in response to a query. The time to access a randomly chosen page stored on a hard disk requires about 10 ms. This is several orders of magnitude slower than retrieving data from main memory. There are four common ways to reduce the cost of I/Os between main and secondary memory: indexing, buffering, parallelism and clustering.

A. Materialized views and indexes are physical structures for accelerating data access that are casually used in data warehouses. However, these data structures generate some maintenance overhead. They also share the same storage space.

B. Buffering a table improves the performance when accessing the data records contained in the table. The table buffers reside locally on each application server in the system. The data of buffered tables can thus be accessed directly from the buffer of the application server. This avoids the time-consuming process of accessing the database. Buffering is also essential to reducing the number of disk I/Os by keeping data pages in main memory.

C. Databases are growing increasingly large. Large volumes of transaction data are collected and stored for later analysis. Multimedia objects like images are increasingly stored in databases. Large-scale parallel database systems increasingly used for:

1) Storing large volumes of data.

2) Processing time-consuming decision-support queries.

3) Providing high throughput for transaction processing.

4) Data can be partitioned across multiple disks for parallel I/O which reduce the time required to retrieve relations from disk.

D. Clustering is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called clusters. Clustering, in the context to databases, refers to the ability of several servers or instances to connect to a single database. An instance is the collection of memory and processes that interacts with a database, which is the set of physical files that actually store data. Clustering offers two major advantages, especially in high-volume database environments:

1) Fault Tolerance: Because there is more than one server or instance for users to connect to, clustering offers an alternative, in the event of individual server failure.

2) Load Balancing: The clustering feature is usually set up to allow users to be automatically allocated to the server with the least load.

A cluster consists of two or more independent, but interconnected, servers. Several hardware vendors have provided cluster capability over the years to meet a variety of needs. Some clusters were intended only to provide high availability by allowing work to be transferred to a secondary node if the active node fails. Others were designed to provide scalability by allowing user connections or work to be distributed across the nodes. Another common feature of a cluster is that it should appear to an application as if it were a single server. Similarly, management of several servers should be as similar to the management of a single server as possible. Clustering is mainly needed to organize the results provided by a search engine. Clustering can also be viewed as a special type of classification. The clusters formed as a result of clustering can be defined as a set of like elements. But the elements from different clusters are not alike. Clustering takes different forms, depending on how the data is stored and allocated resources.

In the remainder of this paper we describe the issues faced when designing an automatic and dynamic database clustering technique for relational databases. And the performance of the various clustering algorithms is compared based on the time taken to form the estimated clusters. A comparative study of clustering algorithms across two different data items is performed here

II. ISSUES IN AUTOMATIC DATABASE CLUSTERING

A. *The data to be clustered should be known.*

A cluster is a collection of commodity components to provide scalability and availability at a low cost. With this in mind, it is possible to create a database cluster for high-end enterprise applications by storing and processing information on commodity nodes. The architecture for a clustered database is distinguished by how data responsibilities are shared among computer nodes. The first issue that arises when designing a clustering technique is what to cluster. Should the entire database be clustered or only parts of it? This question becomes even more critical in the case of dynamic clustering since re-clustering the entire database can be extremely costly. In some cases, only n most accessed objects of a class are clustered. The major issue lies in the fact that how large this n factor is? Should it be fixed or variable and change with each re-clustering based on access frequency? Should it be a percentage of the database? Should the same value n be applied to all clusters/classes or should each cluster have a different value based on its access frequency?

For this, an attribute clustering method which is able to group attributes based on their interdependence so as to mine meaningful patterns is devised. Given a relational table, a conventional clustering algorithm groups tuples, each of which is characterized by a set of attributes, into clusters based on similarity. But here we present a methodology to group attributes that are interdependent or correlated with each other. Attribute clustering is able to reduce the search dimension.

B. Clustering Method should be known.

The next issue is how to cluster/re-cluster. Traditional database clustering groups together objects in the database based on some similarity criteria. Database clustering can take place along two dimensions: attribute (vertical) clustering and record (horizontal) clustering. Attribute clustering groups together attributes from different database relations while record clustering groups together records from different database relations. When the clustering takes place along both dimensions, the clustering is said to be mixed. A special kind of database clustering is database partitioning (or database fragmentation) where the grouping is performed within each database relation instead of between database relations. *"Traditional Database Clustering"* looks for similarities in the metadata such as the co-access frequencies to group objects, i.e. objects that are accessed together are grouped together whereas *"Data Mining Clustering"* typically looks for similarity in the actual data to group data objects based on some distance function. *"Attribute clustering"* refers to traditional attribute clustering which generates attribute clusters (also called vertical clusters/partitions/fragments in the literature) and *"Record Clustering"* refers to traditional record clustering which generates record clusters (also called horizontal clusters/partitions/ fragments in the literature). In attribute clustering, attributes of a relation are divided into groups based on their affinity. Clusters consist of smaller records, therefore, fewer pages from secondary memory are accessed to process transactions that retrieve or update only some attributes from the relation, instead of the entire record. This leads to better performance. The problem with attribute-clustered databases is that only attribute access frequency, not record frequency, is considered. Thus data records needed to answer a frequent query could be scattered at random across multiple disk blocks. A good clustering technique should be mixed and cluster along both dimensions. Data mining clustering algorithms use a distance measure to compute the distance between any two data objects' values. Data objects are then assigned to clusters such that the distance between objects within a cluster is less than a given threshold and the distance between objects in different clusters is greater than a given threshold. As an example, the BIRCH algorithm (Zhang, 1996) creates a tree of clusters such that all objects in a cluster are no further than a given distance from the center of the cluster. New objects are added to clusters by descending the tree and according to the same criteria. When clusters reach a certain number of objects they are split into two sub-clusters. The process continues until all objects belong to a cluster.

C. Which Clustering Technique to be used: Dynamic Vs. Static?

Another design issue is whether the clustering is static or dynamic. Clustering techniques can be labeled as static or dynamic. With static clustering, data objects are assigned to a disk block once at creation time, then, their locations on disk are never changed. There are three problems with that approach. First of all, in order to obtain good query response time, it requires that the DBA know how to cluster data efficiently at the time the clustering operation is performed. This means that the system must be observed for a significant amount of time until queries and data trends are discovered before the clustering operation can take place. This, in turn, implies that the system must function for a while without any clustering. Even then, after the clustering process is completed, nothing guarantees that the real trends in queries and data have been discovered. Thus the clustering result may not be good. In this case, the database users may experience very long query response time. In some dynamic applications queries tend to change over time and a clustering scheme is implemented to optimize the response time for one particular set of queries. Thus, if the queries or their relative frequencies change, the clustering result may no longer be adequate. The same holds true for a database that supports new applications with different query sets.

D. Triggering the re-clustering process

Whenever we start any application in our system, it affects the overall performance of the system. Invoking the CA too often would impact the system performance because a lot of CPU time would be lost performing unnecessary calculations. So the most important issue in automatic clustering is when to trigger the Cluster Analyzer.

One solution would be to trigger the CA when query response time drops below a user-defined threshold. Now how this threshold can be calculated? Will it be constant or changing with the load on the database or the number of queries being fired on the database? So the ultimate goal is reduce the number of false positives by finding a way to reduce the number of times the CA is triggered. And this threshold is variable based on the number of queries being fired on the database. For this statistics need to be collected. So the time when there is less load on the database, CA should be triggered.

E. Statistics to be collected

To effectively diagnose a performance problem, it is vital to have an established performance baseline for later comparison when the system is running poorly. Without a baseline data point, it can be very difficult to identify new problems. For example, perhaps the volume of transactions on the system has increased, or the transaction profile or application has changed, or the number of users has increased. Collecting statistics is a costly operation, not only it requires a lot of work and processing time but also it can require a lot of memory usage. Another problem related to collecting statistics is how much statistics is enough? When to stop collecting and how to remove the noise in the data? Some techniques such as DSTC actually filter the statistics collected before triggering the CA.

III. DATABASE CLUSTERING ALGORITHMS

Clustering is an important application area for many fields including data mining, statistical data analysis, compression, vector quantization and other business applications. Various clustering algorithms have already been proposed. Few of them are discussed here.

1) *Partitioning Algorithm:* Construct various partitions then evaluate them by some criterion (CLARANS, O(n) calls)
2) *Hierarchy Algorithm:* Create a hierarchical decomposition of the set of data (or objects) using some criterion (merge & divisive, difficult to find termination condition).

<u>Algorithm1: K-means clustering</u>

A. What is K-means clustering?

Clustering is the process of partitioning a group of data points into a small number of clusters. For instance, the items in a supermarket are clustered in categories (butter, cheese and milk are grouped in dairy products). Of course this is a qualitative kind of partitioning. A quantitative approach would be to measure certain features of the products, say percentage of milk and others, and products with high percentage of milk would be grouped together. In general, we have n data points xi,i=1...n that have to be partitioned in k clusters. The goal is to assign a cluster to each data point. K-means is a clustering method that aims to find the positions μi,i=1...k of the clusters that minimize the distance from the data points to the cluster. K-means clustering solves:

$$\arg\min_{\mathbf{c}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in c_i} d(\mathbf{x},\mu_i) = \arg\min_{\mathbf{c}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in c_i} \|\mathbf{x}-\mu_i\|_2^2$$

where ci is the set of points that belong to cluster i. The K-means clustering uses the square of the Euclidean distance d(x,μi)=‖‖x−μi‖‖22. This problem is not trivial (in fact it is NP-hard), so the K-means algorithm only hopes to find the global minimum, possibly getting stuck in a different solution.

B. K-means algorithm

The Lloyd's algorithm, mostly known as k-means algorithm, is used to solve the k-means clustering problem and works as follows. First, decide the number of clusters k. Then:

1. Initialize the center of the clusters	μi= some value ,i=1,...,k
2. Attribute the closest cluster to each data point	ci={j:d(xj,μi)≤d(xj,μl),l≠i,j=1,...,n}
3. Set the position of each cluster to the mean of all data points belonging to that cluster	μi=1|ci|∑j∈cixj,∀i
4. Repeat steps 2-3 until convergence	
Notation	|c|= number of elements in c

The algorithm eventually converges to a point, although it is not necessarily the minimum of the sum of squares. That is because the problem is non-convex and the algorithm is just a heuristic, converging to a local minimum. The algorithm stops when the assignments do not change from one iteration to the next.

C. Deciding the number of clusters

The number of clusters should match the data. An incorrect choice of the number of clusters will invalidate the whole process. An empirical way to find the best number of clusters is to try K-means clustering with different number of clusters and measure the resulting sum of squares.

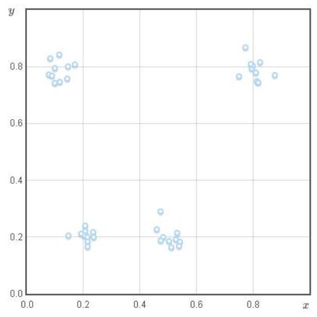
D. Initializing the position of the clusters

It is really up to you! Here are some common methods:

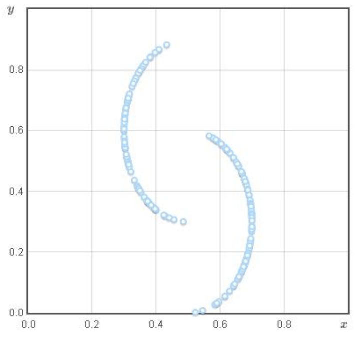
1)Forgy: set the positions of the k clusters to k observations chosen randomly from the dataset.

2)Random partition: assign a cluster randomly to each observation and compute means as in step 3.

Since the algorithm stops in a local minimum, the initial position of the clusters is very important.



Good Example



Bad Example

<u>Algorithm2: The Density Based SCAN Algorithm</u>

The DBSCAN algorithm can identify clusters in large spatial data sets by looking at the local density of database elements, using only one input parameter. Furthermore, the user gets a suggestion on which parameter value that would be suitable. Therefore, minimal knowledge of the domain is required. The DBSCAN can also determine what information should be classified as noise or outliers. In spite of this, its working process is quick and scales very well with the size of the database – almost linearly. By using the density distribution of nodes in the database, DBSCAN can categorize these nodes into separate clusters that define the different classes. DBSCAN can find clusters of arbitrary shape, as can be seen in figure 1 [1]. However, clusters that lie close to each other tend to belong to the same class.

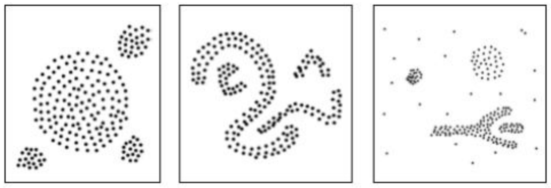


Figure 1. The node distribution of three different databases, taken from SEQUOIA 2000 benchmark database.

1) Density = number of points within a specified radius r (Eps)

2) A point is a core point if it has more than a specified number of points (MinPts) within Eps

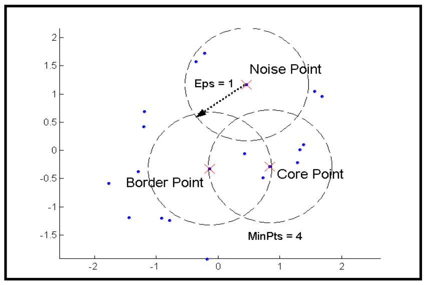
i. These are points that are at the interior of a cluster

3) A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point

4) A noise point is any point that is not a core point or a border point.

<u>Algorithm3: CURE Algorithm</u>

Initially since all points are in separate clusters, each cluster is defined by the point in the cluster. Clusters are merged until they contain at least c points. The first scattered point in a cluster in one which is farthest away from the clusters centroid. Other scattered points are chosen so that their distance from previously chosen scattered points in maximal. When c well scattered points are calculated they are shrunk by some factor α (r = p + α*(mean-p)). After clusters have c representatives the



distance between two clusters is the distance between two of the closest representatives of each cluster.Every time two clusters are merged their representatives are re-calculated.CURE is positioned between centroid based (d_{ave}) and all point (d_{min}) extremes. A constant number of *well scattered* pointsis used to capture the shape and extend of a cluster. The points are *shrunk* towards the centroid of the cluster by a factor α. These *well scattered* and *shrunk* points are used as representative of the cluster. Scattered points approach alleviates shortcomings of d_{ave} and d_{min}.

– Since multiple representatives are used the splitting of large clusters is avoided.
– Multiple representatives allow for discovery of non spherical clusters.
– The shrinking phase will affect outliers more than other points since their distance from the centroid will be decreased more than that of regular points.

IV. CONCLUSION

We have discussed here various issues which need to be solved when designing a database clustering technique. We also presented our framework for an automatic and dynamic mixed database clustering technique. A list of statistics should be gathered. If bad clustering is detected, then we need to trigger a re-clustering process. Then we discussed various clustering algorithms which can be used for designing a clustering technique. As the number of cluster increases gradually, the time to form the clusters also increases. Apart from these there can be efficient ranking based algorithm also. Any of which can be used for designing a clustering technique.

V. Acknowledgements

I would like to place on record my deep sense of gratitude to **Dr. Deepti Sharma,** Head of Department of Computer Science and Engineering, A.I.T.M,Palwal, India for her generous guidance, help and useful suggestions. It is with deep sense of gratitude and reverence that I express my sincere thanks to my supervisor **Mr. Mahesh Chauhan**, Asstt. Prof, AITM for his guidance, encouragement, help and valuable suggestions throughout. His untiring and painstaking efforts, methodical approach and individual help made it possible for me to complete this work in time. I am thankful to **Mr Sunil Sharma**, Asstt. Prof., J B Knowledge Park for constant encouragement and moral support. I am extremely thankful to **Dr. R.S.Chaudhary** Director A.E.I Palwal, for providing me infrastructure facilities to work to work in, without which this work would not have been possible.

REFERENCES

[1] (Agrawal, **2004)** Sanjay Agrawal, Vivek Narasayya, and Beverly Yang, ***"Inreg~nting Vevficol nnd Horizontnl Pnrtilioning into Automafed Physicnl Dntnbose Design,"*** the 2004 ACM SIGMOD lnternational Conference on Management of Data. June 2004.

[2] (AMS, **2003)** Automatic Computing Workshop, ***5"'*** Annual lnternational Wwkshop on Active Middleware Services., June 2003.

[3] (Aouiche, **2003)** Kamel Aouiche, Jerome Darmont, and Le Gruenwald, "Frequent ltemsets Mininig for Database Auto-Administration", the lnternational Database Engineering and Applications Symposium, 2003, 16-18 July 2003, pages 98-1 03.

[4] (Bernstein, **1998)** Phil Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Frankiln, Hector Garcia-Molina, Jim Gray, Jeny Held, Joe Hellerstein, H. V. Jagadish, Michael Lesk, Dave Maier, Jeff Naughton, Hamid Pirahesh, Mike Stonebraker, and Jeff Ullman, "The Asilomar Report on Database Research", ACM SIGMOD Record,Vol. 27 , Issue 4, pp. 74-80, December 1998.

[5] (Brinkhoff, **2001)** Thomas Brinkhoff, "Using a Cluster Manager in a Spatial Database System", proceedings of the ninth ACM international symposium on Advances in geographic information systems, 2001, pp. 136.141.

[6] C. S. Li, “Cluster Center Initialization Method for K-means Algorithm Over Data Sets with Two Clusters”, “2011 International Conference on Advances in Engineering, Elsevier”, pp. 324-328, vol.24, 2011.

[7] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko and A. Wu, “An Efficient K-Means Clustering Algorithm: Analysis and Implementation”, “IEEE Transactions on Pattern analysis and Machine intelligence”, vol. 24, no.7, 2002.

[8] Y.M. Cheung, “A New Generalized K-Means Clustering Algorithm”,“Pattern Recognition Letters, Elsevier”,vol.24,issue15, 2883–2893, Nov.2003.

[9] Z. Li, J. Yuan, H. Yang and Ke Zhang, “K-Mean Algorithm with a Distance Based on the Characteristic of Differences”, ”IEEE International conference on Wireless communications, Networking and mobile computing”, pp. 1-4, Oct.2008.

[10] A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise
Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu,

[11] Advances in knowledge discovery and data mining: 9th Pacific-Asia conference, PAKDD 2005, Hanoi,
Vietnam, May 18-20, 2005; proceedings Tu Bao Ho, David Cheung, Huan Liu

[12] Anomaly Detection in Temperature Data Using DBSCAN Algorithm: Erciyes Univeristy, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5946052&tag=1 Mete Celic, Filiz Dadaser-Celic, Ahmet Sakir DOKUZ