Research Issues in Automatic Database Clustering

Sylvain Guinepain and Le Gruenwald
School of Computer Science
The University of Oklahoma
Norman, OK 73019, USA
{Sylvain.Guinepain, ggruenwald}@ou.edu

Abstract

While a lot of work has been published on clustering of data on storage medium, little has been done about automating this process. This is an important area because with data proliferation, human attention has become a precious and expensive resource. Our goal is to develop 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. In this paper we describe the issues that need to be solved when developing such a technique.

1. 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 (Elmasri, 2003). 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, clustering and parallelism.

Much research has been done on indexing, buffering, clustering and parallelism. Some attempts to automate the indexing process have been undertaken (Chaudhuri, 1998; Aouiche, 2003). Several researchers have also worked on automating clustering (Brinkhoff, 2001; Darmont, 2000; Gay, 1997; McIver, 1994; Zaman, 2004), but these techniques are not fully automated and require lots of parameters and users' hints.

Few techniques have been designed to cluster large databases on storage medium. In our opinion, clustering is as important as indexing for a good clustering technique can dramatically reduce the number of I/Os, which will in turn speed up query response time. As noted in (Omiecinski, 1990), record clustering can be viewed as a complementary problem to indexing. Indexes greatly improve complex queries' response time by identifying the records that are required. However if the data records themselves are not clustered into as few disk pages as possible, many disk accesses will be needed. Thus, an appropriate index will reduce the number of records to be retrieved while clustering these records on the same or adjacent pages will ensure that the number of disk accesses is minimized.

Parallelism spreads data across multiple disks so that it can be retrieved in parallel (Ferhatosmanoglu, 1999). However, de-clustering techniques distribute data on different disks but do not specify the in-disk organization of data within a single disk. The following quote from the Asilomar Report on Database Research (Bernstein, 1998) suggests that besides exploiting parallelism, careful organization of each disk is essential for efficient retrieval:

> *"While disk capacity is improving very quickly, seek times are improving relatively slowly. Hence, the amount of data that can be transferred to main memory during an average seek time is rising very quickly. Put differently, the cost of a seek relative to the transfer time of a byte of data is rising quickly."*

Buffering is also essential to reducing the number of disk I/Os by keeping data pages in main memory. Using a page replacement strategy such as LRU (Least Recently Used) the buffer manager attempts to keep in memory data pages that may be accessed in the near future, hence, eliminating some disk I/Os. However if the underlying physical clustering scheme is not good enough, the buffer manager will become powerless for all data records needed in response to a query may reside on different disk pages. If some of these pages are not in the buffer at the time of the query, the system will have to perform additional disk I/Os.

Therefore, 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.

A distinction among clustering techniques is whether they are manual or automatic. A manual clustering technique has to be started by the user. This in turn implies that the user should know when to run the re-clustering process. Even if the user developed a good feeling for when he or she should run the re-clustering, it would still be a great inconvenience, especially in a very dynamic environment where re-clustering could be triggered frequently. Rather, we advocate using an intelligent automatic clustering that could trigger itself when most appropriate. Another argument in favor of auto-clustering is the prediction from the Asilomar report on database research (Bernstein, 1998) that in the near future everything will be monitored through the Internet and that trillions of gizmos will need billions of servers. Because of this, the report concludes the following:

> *"The relative cost of computing and human attention has changed: human attention is the precious resource. This new economics requires that computer systems be autoeverything: autoinstalling, automanaging, autohealing, and autoprogramming. Computers can augment human intelligence by analyzing and summarizing data, by organizing it, by intelligently answering direct questions and by informing people when interesting things happen."*

In (Aouiche, 2003) and (Zaman, 2004), the authors describe an attempt to automate database indexing to reduce human intervention. Microsoft addresses the same problem in (Chaudhuri, 1998) and later addresses the problem of integrating vertical and horizontal partitioning into automated physical database design in (Agrawal, 2004). The area of automatic computing has been getting a lot of attention lately as several conferences are dedicated to the topic (SAACS, 2004; AMS, 2003). In (Dolev, 2004) the authors explain how to design self-stabilizing operating systems. Twenty years after the movie "War Games" (1983) directed by John Badham, (Ibrahim, 2004) discuses the issues involved with removing the man from the loop and describes the systems where such a move is possible.

In the remainder of this paper we describe the issues faced when designing an automatic and dynamic database clustering technique for relational databases. In Section 2 we first review the challenges solved by traditional database clustering methods, then we discuss the issues encountered when designing an automatic and dynamic clustering technique. Whenever possible we also include in the discussion the way we approach the problem in our own automatic clustering technique currently under development, called AutoClust. Finally in Section 3 we give our conclusions.

2. Issues in Automatic Database Clustering

2.1 What to cluster?

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. Some techniques like StatClust (Gay, 1997) only re-cluster the *m* most accessed objects of each class. The problem then is to determine how big the parameter *m* should be. 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 *m* be applied to all clusters/classes or should each cluster have a different value based on its access frequency?

In AutoClust, we propose that attribute clusters be chosen based on frequent closed item sets (Pasquier, 1999; Durant, 2002). A closed item set is a maximal item set contained in the same transactions. For instance, if attributes A, C, and F form a frequent closed item set for a given support level threshold, then {A, C, F} will be considered as an attribute cluster. Re-clustering the entire database can be very costly. Instead, AutoClust could re-cluster only attribute clusters having a support level greater than a user-defined threshold or re-cluster each cluster proportionally to its support level. For instance, an attribute cluster with a support level of 30% would have the 30% most frequently accessed tuples re-clustered. The advantage of this last solution is that it removes the need for a user parameter and moves us one step closer to a fully automated solution.

2.2 How to cluster?

The next issue is how to cluster/re-cluster. Database clustering has been an area of research for decades, foundation papers can be retraced as far back as the early 1970's (McCormick, 1972).

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 (Silberschatz, 2002). However, the term "*database clustering*" has been used loosely in the literature and is sometimes used in place of "*database partitioning*" (Yu, 1985; Omiecinski, 1990).

Another possible confusion may occur between traditional database clustering and data mining clustering. While both have the same objective of grouping together objects based on some similarity criteria, it is how they achieve this goal that differentiates them. 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. Instead, data mining clustering typically looks for similarity in the actual data to group data objects based on some distance function. Objects that have similar data values are grouped together independently of whether they are accessed together or not. In the remainder of this paper we use "*database clustering*" to refer to traditional database clustering and "*data mining clustering*" for its data mining counterpart. "*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 the remainder of this section we review the literature in database clustering/partitioning as well as data mining clustering.

Many techniques have been designed for record clustering for relational databases (Yu, 1985; Omiecinski, 1990), for object-oriented databases (Hudson, 1989; Chang, 1989; Kim, 1990, McIver, 1994, Gay, 1997, Darmont, 2000), and attribute clustering and partitioning (McCormick, 1972; Navathe, 1984; Navathe, 1989; Chu, 1993; Hartuv, 2000). With record clustering, relations are broken down into groups of records based on their affinity. Records that are more frequently used together are placed in the same groups. These groups are then assigned to physical pages. The most frequently accessed groups can also be assigned to faster memories. The problem with record clustering is that not all queries require all attributes of a record. In fact, some attributes may never be queried at all. Thus, when retrieving records from a record-clustered database, some of the retrieved information is useless leading to a poor performance. Attribute clustering helps solving this problem.

Most record clustering techniques use some kind of statistical analysis to cluster records together. In (Yu, 1985) the authors assign a position line to each record. After each query, the records accessed are then reassigned a position line closer to the centroid for that query. The idea is that eventually the records queried together will converge within the same location in the data file. In (Omiecinski, 1990) the authors formulate the record clustering problem as minimizing the objective function: $C = \sum_{i=1}^{M} F(Q_i) * P(Q_i)$ where $F(Q_i)$ is the frequency of query Q_i and $P(Q_i)$ is the number of pages which contain records for query Q_i. Cactis (Hudson, 1989) is a clustering algorithm for object-oriented databases that stores objects based on their co-access frequencies. The most frequently accessed object is stored in a new block along with all the objects that are most frequently accessed with it. The process is repeated page after page until all objects are stored. ORION (Kim, 1990) stores objects along with their composite hierarchy. This technique targets applications where objects are queried along with their hierarchy. In (McIver, 1994) the clustering process uses two metrics, simple object references (heat) and co-references (tension). The heat of an object is the frequency with which it has been accessed. The tension of a pair of objects expresses the likelihood that the two objects will be accessed together over the course of a series of transactions. The likelihood that a pair of objects will be accessed together is what will determine whether or not they should be stored together on disk. (Gay, 1997) uses four kinds of statistics (inter-class relationship, read/write ratio at the class level, access count for each individual object and statistics about the buffering process) to cluster objects. DRO (Darmont, 2000) uses two types of statistics: the object access frequency and the page usage rate which help identify pages that degrade the system performance.

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 (Navathe, 1984). 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.

The attribute clustering problem is a very complex problem and the number of possible solutions is equal to the Bell number that satisfies the following recurrence relation: $b_{n+1} = \sum_{k=0}^{n} b_k \binom{n}{k}$.

The Bond Energy Algorithm (BEA) (McCormick, 1972) is used to cluster database attributes. It creates an *NxN* matrix *A* where *N* is the number of attributes. The intersection of row *i* and column *j* contains the co-access

frequency between attributes *i* and *j*. The algorithm then minimizes the value of $ME(A) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} [a_{i,j+1} + a_{i,j-1} + a_{i+1,j} + a_{i-1,j}]$, where $a_{0,j} = a_{M+1,j} = a_{i,0} = a_{i,N+1} = 0$ and *A* is a nonnegative *MxN* array by permuting the rows and columns of *A*. In the end, *A* is in block diagonal form and each block of attributes can be used as an attribute cluster.

In NVP (Navathe, 1984), the authors use the output of the BEA algorithm, which is a block diagonal matrix. The algorithm then finds the best location for a point *x* along the diagonal of the *CA* matrix. The point *x* splits the matrix attributes into two clusters. This splitting process is repeated until the resulting clusters minimize an objective function. In (Navathe, 1989), the authors propose a solution to the attribute clustering problem based on graph theory. A graph is created where each node represents an attribute and edges are weighted using the affinity values between attributes. Nodes/attributes that form a primitive cycle in the graph are clustered together. A proof is given that the solution is not dependent on the starting node. In the Optimal Binary Partitioning algorithm (Chu, 1993) the authors use transactions to split the set of attributes into two subsets and discover the optimal binary partition of the set of attributes. (Hartuv, 2000) proposes a clustering technique based on graph connectivity that aims at partitioning gene expression data in the field of bio-informatics. This technique is applicable to either attribute or record clustering. It is graph theoretic. The similarity data is used to form a similarity graph in which vertices correspond to elements and edges connect elements with similarity values above some threshold. In that graph, clusters are highly connected sub-graphs defined as sub-graphs, the edge connectivity of which exceeds half of the number of vertices.

While in traditional database clustering, objects are grouped based on similarity in access patterns, in data mining clustering, objects are clustered based on similarity in the actual data. The more similar two objects are, the more likely they belong to the same cluster (Dunham, 2004). 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.

AutoClust is a mixed database clustering technique. First, the attribute clustering is done by mining frequent closed item sets using existing algorithms such as A-CLOSE (Pasquier, 1999) or CHARM (Zaki, 2002). The frequent closed item sets are then fed to an algorithm we have developed that considers all possible attribute clusters containing at least one cluster of attributes belonging to the set of frequent closed itemsets. AutoClust then selects the cluster of attributes that performs the best using its cost model. Within each attribute cluster created, record clustering is then done using a data mining clustering technique such as BIRCH.

2.3 Static clustering vs. dynamic clustering

Another design issue is whether the clustering is static or dynamic. Clustering techniques can be labeled as static or dynamic (Darmont, 1996; Gay, 1997). 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 such as GIS (Brinkhoff, 2001), 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.

In contrast, with dynamic clustering such as in AutoClust, objects are being relocated on disk if it is determined that the clustering in place has become inadequate due to a change in query patterns or database size.

The remaining issues discussed in Sections 2.4-2.7 deal with the automatic aspect of the clustering. Most automatic clustering techniques (McIver, 1994; Gay, 1997; Darmont, 2000) consist of the following modules: a Statistic Collector (SC) that accumulates information about the queries run and data returned. The SC is in charge of collecting, filtering, and analyzing the statistics. It is responsible for triggering the Cluster Analyzer (CA). The CA determines the best possible clustering given the

statistics collected. If the new clustering is better than the one in place, then the CA triggers the reorganizer that physically reorganizes the data on disk.

2.4 When to trigger the re-clustering process?

The most important issue in automatic clustering is when to trigger the Cluster Analyzer. Invoking the CA too often would impact the system performance because a lot of CPU time would be lost performing unnecessary calculations. On the other hand, not invoking the CA often enough would also negatively impact the system by letting queries run against an obsolete clustering scheme. Careful consideration must then be given to the problem of when to trigger the CA.

One solution would be to trigger the CA when query response time drops below a user-defined threshold. The sub-issues would then be how to set the threshold and whether that threshold is a constant or variable. In StatClust (Gay, 1997) the SC collects statistics until it has established using confidence interval that the statistics collected is meaningful. The problem with that mathematically elegant solution is that the CA triggering is not related to the system performance, and the CA is likely to be triggered even if the query response time is adequate. Other alternatives would be to trigger the CA when there is too much time between re-clustering or when too many queries have run, too many data items have been queried or too many records have been accessed. Once again these solutions do not use the system response time and, therefore, could trigger unnecessary reorganizations.

The ultimate goal is reduce the number of false positives by finding a way to reduce the number of times the CA is triggered. In AutoClust we intend to use the query response time as a criterion to trigger the CA. Whenever the query response time drops below a threshold, the CA is triggered. To fully automate our technique and to avoid human intervention, the threshold is variable. For instance, if the queries tend to get longer and the users' given threshold cannot be satisfied, our adaptive algorithm progressively augments the threshold.

2.5 What statistics to collect?

Since most automatic clustering techniques collect statistics to achieve their goals, a major issue is what statistics to collect. (Omiecinski, 1990) collects for each query, the frequency of occurrence of the query and the locations of the records that satisfy the query. Cactis (Hudson, 1989) keeps track of the number of times each database object is accessed as well as the number of times each relationship between objects in the process of evaluation or marking out-of-date is crossed. (McIver, 1994) keeps track of the heat and tension between objects. In StatClust (Gay, 1997), statistics about inter-class relationships are collected as well as the read/write access ratio at the class level, the number of accesses for each object and some statistics about the buffering process. DSTC and DRO (Darmont, 2000) keep track of the object access frequency and the page usage rate.

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.

AutoClust does not collect any statistics. Instead it uses a query log that contains for each query, the attributes accessed, the number of tuples returned and the number of disk blocks accessed. AutoClust mines the frequent closed item sets using the attributes as items.

2.6 How to detect bad clustering?

Another critical issue is how to detect bad clustering. We distinguish two kinds of bad clustering, namely record and attribute. Most techniques detect bad record clustering by computing the ratio between the number of disk pages accessed and the number of records retrieved or between the number of records retrieved in the memory buffer and the total number of records retrieved. So a common technique as far as detecting bad record clustering seems to be to detect that the number of disk pages accessed is too high compared to a given threshold. This threshold, once again, may have to be provided by the user as an initial parameter and may have to vary over time.

Though initially it seems that the same technique could be applied to detect bad attribute clustering, we believe there is a major difference between bad record clustering and bad attribute clustering. The former means that too many disk pages are being accessed whereas the latter means that too many clusters are being accessed. Therefore we advocate that bad attribute clustering should be detected when the number of clusters accessed is greater than some user given threshold. Note that if the number of clusters accessed is too high then so will the number of disk pages. Thus, bad attribute clustering is likely to cause bad record clustering as well but the reverse is not always true. For this reason, we recommend testing for bad record clustering before testing for bad attribute clustering since we could have bad record clustering without having bad attribute clustering. In addition, bad attribute clustering is a more severe problem than bad record clustering but it is also less frequent.

2.7 How to re-cluster?

If attribute re-clustering is needed, it should take place before record re-clustering because record re-clustering is always required after attribute re-clustering. Therefore, we should first create clusters of attributes, and then we group the records within each attribute cluster to form record clusters. The record-clustering algorithm used does not need to be the same for each attribute cluster. It would be a good idea to select the record-clustering algorithm for an attribute cluster based on the data present in that cluster and the queries run against it. The automatic clustering framework developed should, therefore, facilitate the addition or substitution of record clustering algorithms.

Another very important issue when performing automatic re-clustering is to choose a clustering/re-clustering algorithm that is efficient not only in terms of the quality of the clusters produced but also in terms of speed of execution. Thus, it is a good idea to look into algorithms that are incremental, i.e. those that reuse the results from the previous reorganization to reduce the number of calculations needed for the next re-clustering.

3. Conclusions

In this paper we discussed the issues that need to be solved when designing a database clustering technique. We also presented our framework for an automatic and dynamic mixed database clustering technique currently under development called AutoClust. AutoClust mines closed item sets to create clusters of attributes and uses data mining clustering to perform record clustering within each attribute cluster. AutoClust is triggered when a drop in the query response time is detected. It then checks for bad record and attribute clustering which are detected by an increased number of accesses to record and attribute clusters, respectively. If bad clustering is detected, AutoClust will trigger a re-clustering process.

4. References

(Agrawal, 2004) Sanjay Agrawal, Vivek Narasayya, and Beverly Yang, "*Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design*," the 2004 ACM SIGMOD International Conference on Management of Data. June 2004.

(AMS, 2003) Automatic Computing Workshop, 5th Annual International Workshop on Active Middleware Services., June 2003.

(Aouiche, 2003) Kamel Aouiche, Jerome Darmont, and Le Gruenwald, "Frequent Itemsets Mininig for Database Auto-Administration", the International Database Engineering and Applications Symposium, 2003, 16-18 July 2003, pages 98-103.

(Bernstein, 1998) Phil Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Frankiln, Hector Garcia-Molina, Jim Gray, Jerry 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.

(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.

(Chang, 1989) E. E. Chand and R. H. Katz, "Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering inObject-Oriented DBMS", ACM SIGMOD International Conf. on Data Management, June 1989.

(Chaudhuri, 1998) Surajit Chaudhuri and Vivek Narasayya, 'AutoAdmin "What-if" Index Analysis Utility", SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 1998.

(Chu, 1993) Wesley W. Chu and Ion Tim Ieong, "A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems", IEEE Transactions on Software Engineering, Vol. 19, No. 8, August 1993.

(Darmont, 2000) J. Darmont, C. Fromantin, S. Regnier, L. Gruenwald, M. Schneider, "Dynamic Clustering in Object-Oriented Databases: An Advocacy for Simplicity", ECOOP 2000 Symposium on Objects and Databases, June 2000; LNCS, Vol. 1944, 71-85.

(Dolev, 2004) Shlomi Dolev and Reuven Yagel, "Towards Self-Stabilizing Operating Systems", DEXA 2004, pp. 684-688, Sept. 2004.

(Dunham, 2004) Margaret H. Dunham, "Data Mining, Introductory and Advanced Topics", Prentice Hall, 2004.

(Durand, 2002) Nicolas Durand and Bruno Cremilleux, "Extraction of a Subset of Concepts from Frequent Closed Itemset Lattice: A New Approach of Meaningful Clusters Discovery", 2002.

(Elmasri, 2003) Ramez Elmasri and Shamkant B. Navathe, "Fundamentals Of Database Systems", Addison-Wesley, 2003.

(Ferhatsomanoglu , 1999) Hakan Ferhatsomanoglu, Divyakant Agrawal, Amr El Abbadi, "Clustering Declustered Data for Efficient Retrieval", the Conference on Information and Knowledge Management, Nov. 1999, pages 343--350,

(Gay, 1997) Jean Yves Gay and Le Gruenwald, "A Clustering Technique for Object-Oriented Databases", the 8th International Conference, DEXA '97, September 1997.

(Hartuv, 2000) Erez Hartux, and Ron Shamir, "A Clustering Algorithm Based on Graph Connectivity", Information Processing Letters, Vol. 76, No. 4-6, pp. 175-181, 2000.

(Hudson, 1989) Scott E. Hudson and Roger King, "CACTIS: A Self-Adaptive, Concurrent Implementation of an Object-Oriented Database Management System", ACM Transactions on Database Systems, Vol.14, No.3, Sept. 1989, pp. 291-321.

(Ibrahim. 2004) Mohamed T Ibrahim, Ric Telford, Petre Dini, Pascal Lorenz, Nino Vidovic, and Richard Anthony, "Self Adaptability and the Man-in-the-Loop: A Dilema in Automatic Computing Systems", DEXA 2004, Sept. 2004, .pp. 722-729,.

(Kim, 1990) W. Kim, J. F. Garza, N. Ballou and D. Woelk, "Architecture of the ORION next-generation database system", IEEE Transaction on Knowledge and Data Engineering, Vol. 2, No. 1, 1990.

(McCormick, 1972) McCormick, W. T. Schweitzer P. J., and White T. W., "Problem decomposition and data reorganization by a clustering technique", Oper. Res. 20, 5, September 1972, pp 993-1009.

(McIver, 1994) William J. McIver, Jr. and Roger King, "Self-Adaptive, On-Line Reclustering of Complex Object Data", SIGMOD 94.

(Navathe, 1984) Shamkant Navathe, Stefano Ceri, Gio Wierhold, and Jingle Dou, "Vertical Partitioning Algorithms for Database Design", ACM Transactions on Database Systems, Vol. 9, No. 4, December 1984, pages 680-710.

(Navathe, 1989) Shankant B. Navathe and Minyoung Ra, "Vertical Partitioning for Database Design: A Graphical Algorithm", ACM SIGMOD International Conference on Management of Data, 1989, .pp. 44-450,

(Omiecinski, 1990) Edward Omiecinski and Peter Sheuermann, "A Parallel Algorithm for Record Clustering", ACM Transactions on Database Systems, Vol. 15, No. 4, December 1990, pp. 599-624.

(Pasquier, 1999) Nicolas Pasquier, Yves Bastidem Rafik Taouil, and Lofti Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices", Information Systems, Vol. 24, No. 1, pp. 25-46, 1999.

(SAACS, 2004) 2nd International Workshop on Self-Adaptable and Automatic Computing Systems, DEXA 2004.

(Silberschatz, 2002) Avi Silberschatz, Henry Korth, and S. Sudarshan, "Database System Concepts", 4th edition, McGraw Hill, 2002.

(Yu, 1985) C. T. Yu, Cheing-Mei Suen, K. Lam, and K. Siu, "Adaptive Record Clustering", ACM Transactions on Database Systems, Vol. 10, No. 2, June 1985, pp. 180-204.

(Zaman, 2004) Mujiba Zaman, Jyotsna Surabattula, and Le Gruenwald, "An Auto-Indexing Technique for Databases Based on Clustering", DEXA, Sept. 2004, pp. 776-780,.

(Zhang, 1996) Tian Zhang, Raghu Ramakrishnan, and Miron Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", ACM SIGMOD International Conference on Management of Data, pages 103--114, 1996.