3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)
Proceedings of Engineering & Technology (PET)
pp. 714-720

Contribution to the Query Optimization in Large Databases

Jaafar Noussaiba[*1], Hassen Fadoua[#2], Grissa Touzi Amel[#3]

[*]*Signal, Image and Technology of Information Laboratory, National Engineering School of Tunis, Tunis El Manar University, BP. 37, Le Belvedere 1002, Tunis, Tunisia*

[1]noussaiba.jaafar@gmail.com

[#]*Computer, Programming, Algorithmic and Heuristic Laboratory, Faculty of Mathematical, Physical and Natural Sciences of Tunis, El Manar University Campus, 2092 El Manar, Tunis, Tunisia*

[2]hassen.fadoua@gmail.com

[3]grissa.touzi@topnet.tn

Abstract*— In this paper, we propose a contribution to the query optimization in large databases (DB). Indeed, it is now very important to optimize researches and accelerate queries because of the information volume handled in very large databases and the operations complexity. The indexing technique is probably one of the most used query optimization techniques. However, the modeling and the management of indexes require a large memory space and an update overhead. Therefore, this technique becomes more and more complex in the case of large data volume. We propose in this paper a new query optimization approach based on the indexing and the classification concepts to make the querying results of users faster and provide satisfactory answers. This consists in indexing the data groups obtained following a clustering algorithm by using Data Mining. To validate our approach, we used Oracle as an example of Database Management System (DBMS).

Keywords*— Query Optimization; Large Databases; Indexing; Classification; Clustering; Data Mining.

I. INTRODUCTION

The indexing interest is to research and acquire data increasingly numerous and complex. Creating indexes improves significantly the time for data access by creating direct access paths. Several techniques have been proposed in this context, we can cite the one-dimensional indexes and the multidimensional indexes. Unfortunately, these proposals suffer from the large memory space and the cost management in terms of index update.

In this paper, we propose a new query optimization approach in large databases based on the indexing and the clustering concepts. This approach should allow: 1) an optimal search data across the queries, 2) a possible implementation whatever the type of DBMS, indexing and clustering techniques, 3) a choice of the clustering algorithm based on the data field and the user requirements, 4) possible modifications according to the type of DBMS. This approach has been validated with the Oracle DBMS.

This paper is organized as follows: Section 2 presents the basic concepts of query optimization in DB, indexing and its different techniques, and classification with its various methods especially the clustering methods. Section 3 is devoted to the presentation of our proposed approach. Section 4 presents our tool developed for the query optimization in large DB. Section 5 presents the evaluation of experimental results. We end with conclusions and some perspectives.

II. BASIC CONCEPTS

A. Query Optimization in Databases

Optimizing a query is to determine for this query an optimum execution plan or scenario leading to minimal processing time. It is possible to have several execution scenarios for a given query. Therefore, the optimization is the choice of the best execution plan.

The optimization is realized by an essential component in relational DBMS, which is the optimizer. Indeed, it transforms a query expressed in source language that is the Structured Query Language (SQL) in an implementation plan composed of a sequence of basic operations expressed in a target language and called ‘’low-level’’ effectively performing the data access [1].

Query optimization is often divided into two phases: the logical optimization which can rewrite the query in a simplified canonical form and logically optimized, that is to say without taking into consideration the cost of access to data, and the physical optimization that performs better algorithms for low-level operators considering the data size and the available access paths.

To optimize queries, there are data access methods such as methods with indexing.

B. Indexing

1) *Definitions:* The index is a data structure which optimizes queries to search and acquire information on the database. Indeed, this is an auxiliary file, structured that makes the access to certain data more efficient depending on an index key. Furthermore, the index is a sorted system table containing the user data to accelerate the processing.
2) *Indexing techniques:* Several indexing techniques have been proposed. We cite some examples of one-dimensional indexes and multidimensional indexes.
   - **One-dimensional indexes:** An index is considered to be one-dimensional if the data are from a single attribute field.

ISSN: 2356-5608

Among the one-dimensional indexes, we cite the hash tables that allow direct access to a specific record t based on a function called hash function f, and the B-Tree [2] which is a linked list of nodes whose value is that of the index. It is a search tree on several levels which is sorted and balanced. The B-Tree offers an excellent compromise for the search operations by key and interval, as well as updates.

- **Multidimensional indexes:** A multidimensional space is defined when the elements of the considered set are homogeneous and heterogeneous vectors whose components are totally ordered. The multidimensional indexing techniques are based on the principle of grouping the basis vectors into packets, then to encompass them in simple geometric shapes to handle. These techniques can be classified into two main approaches: data partitioning and space partitioning.

  Among the techniques of the data partitioning approach, we find the R-Tree [3], [4] which is a spatial access method used to index the geographic coordinates and the SR-Tree [5] that indexes the incorporated areas of the intersection between a hyper-sphere and a hyper-rectangle. These methods suffer from the problem of the curse of dimensionality, that is to say their performances degrade when the dimension increases. The major drawback of these methods is the overlap between the geometric shapes including the vectors.

  The space partitioning approach is based on the constraint where the intersections of the areas are null. The space partitioning methods partition the space data in disjoint geometric shapes. We cite the following techniques: the kD-Tree [6] which is the basic structure of all arboreal indexes where it is based on the partitioning of a k-dimensional space according to each of its axes, and the pyramid-Tree [7] that provides significant performances on disk access and response time. There are several other methods such as TV-Tree [8], Grid-File [9] and BANG-File [10]. These methods use disjoint geometric shapes without overlap, but as the methods based on the data partitioning, they suffer from the problem of the curse of dimensionality where their dimensions are considerably degraded by increasing the dimension.

- **Indexing techniques limitations:**

  Generally, indexes offer several advantages, however they also have limitations. In fact, whatever the used indexing method one-dimensional or multidimensional, we need to store the index in a tree or a table. Therefore, using indexes require a large storage space for their modeling and management especially during updates.

  Regarding the multidimensional indexing techniques, the curse of dimensionality is the major problem affecting the majority of multidimensional indexing methods in large dimension.

C. *Classification*

1) *Definitions:* The classification is a main task of the Data Mining step. In general, the classification consists to group into homogeneous classes a set of objects for descriptive or decision-making purposes. Therefore, this consists to organize a set of similar objects into groups. These groups are the most consistent and homogeneous, and called clusters.
2) *Types:* The classification is usually divided into two groups: supervised classification and unsupervised classification. For supervised classification, it is able to classify a new object from a set of predefined classes by using labeled examples. This approach requires the intervention of a human expert to label information before their classification. In the unsupervised classification methods or clustering, classes are not predefined and possible examples are not labeled. We are interested in clustering.
3) *Clustering methods:* Several clustering methods have been conducted. These methods can be divided mainly into two categories: hierarchical clustering and partitioning clustering.
   - **Hierarchical clustering:** This consists to create a hierarchical decomposition of objects according to certain criteria. Hierarchical methods generate a sequence of nested partitions into each other. The arboreal representation of groups is called a "dendrogram".
   - **Partitioning clustering:** The partitioning algorithms are used to build partitions and evaluate them according to certain criteria. These algorithms provide, as output, a partition of the objects space rather than an organizational structure of dendrogram type. Among the most used algorithms, we cite the k-Means algorithm that is simple, comprehensible, and the elements are assigned automatically to clusters. There are several versions of this algorithm such as PAM (Partitioning Around Medoids) [11] and Fuzzy-k-Means (FCM) [12], [13].

III. NEW PROPOSED APPROACH

We propose a query optimization approach based on two main concepts that are the indexing and the classification. This consists to apply the indexing on the groups obtained after the data classification. Thus, a new concept was born which is the concept of "Meta-Index". Our approach is based on two properties: 1) the number of the groups generated following a classification algorithm is always significantly lower than the initial data 2) all the objects in the same group have the same characteristics.

The general schema of the proposed approach of query optimization is presented as following:

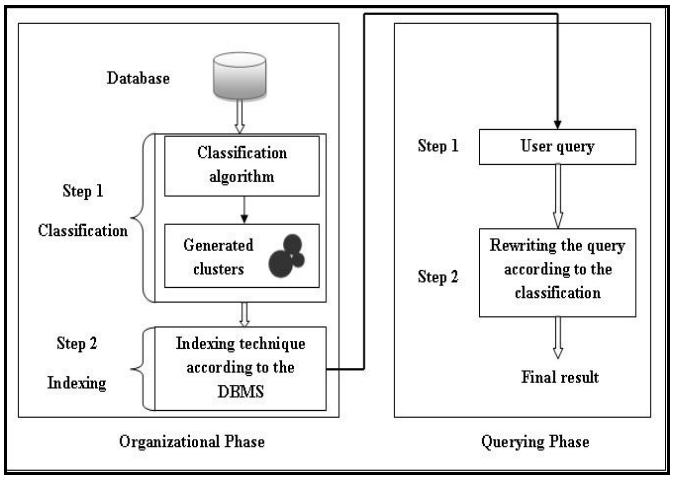

Fig.1 Principle of the proposed approach of query optimization

The query optimization process takes place in two main related phases. The first phase is the organizational phase taking place in two steps which are the classification and the indexing. The second phase is the querying phase which is presented by the user query and its rewriting by using the classification principle.

A. Organizational phase

The data organization is the first phase of our approach. From the clusters generated following a clustering operation on a definite attribute, we create an index on the set of the obtained clusters.

1) *Classification step:* The first step is to apply the classification technique by a clustering algorithm. The idea is to group the data in similar groups according to a specific criterion.
2) *Indexing step:* The second step is presented by applying an indexing technique according to the chosen DBMS. This consists to index the data set which is contained in the generated clusters, and not the initial data set. For this, we apply the indexing technique on the table containing the classified data by choosing the indexing attribute.

B. Querying phase

The querying is the second phase of this approach. From classified and indexed data, we must be able to search for responses to satisfy a specific query. This phase allows rewriting the user query according to the classification result. Therefore, the table becomes the table containing the classified data of generated clusters with the corresponding attribute.

C. Validation

To concretize our approach, we used the Oracle DBMS (Oracle 12c) which includes a Data Mining Tool, offering clustering algorithms. In addition, Oracle offers several index types. We used the B-Tree index and the Enhanced-k-Means clustering algorithm that is available in Oracle Data Mining, to validate our approach. In what follows, we detail this validation in reliance on our database described by the following tables:

Student (<u>stud_id</u>, last_name, first_name, adress, section, year)
Material (<u>mat_code</u>, mat_name, coeff, section)
Exam (<u>num_ex</u>, date_ex, coeff, mat_code #)
Notation (<u>stud_id #, num_ex #</u>, note)

1) Organizational phase:

- **Classification step:** To validate this step, we used the Oracle Data Mining tool for data classification by applying an algorithm among the clustering algorithms proposed by this tool. We choose the Enhanced-k-Means algorithm, which is a hierarchical algorithm based on distance. To facilitate the work, Oracle Data Mining includes Oracle Data Miner, a graphical user interface that allows generating, evaluating, and applying Data Mining models. **Example:** In this example, our goal is to classify students depending on their sections; students belonging to the same section will be performed to the same group or cluster. For this, we proceed as follows: At first, we add the data source (Student table). Then, we explore the data by choosing the classification attribute (Section). After, we build the clusters by selecting the Enhanced-k-Means algorithm and configuring its inputs and parameters. Finally, we apply the algorithm. The following figure describes the different steps of the clustering process with Oracle Data Mining.

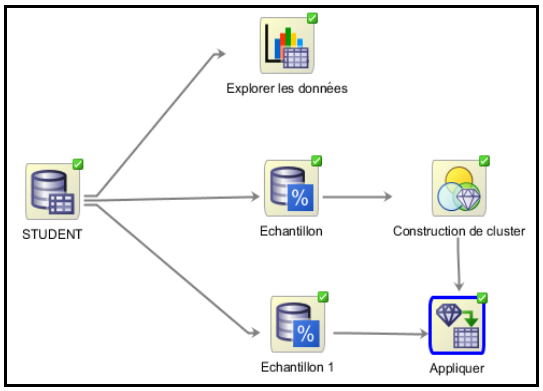

Fig. 2 The different steps of clustering process with Oracle Data Mining

After applying the algorithm, we can visualize the result of the generated clusters. The result is a table (Figure 3) which performs each student to his group by specifying the identifier of each cluster and the membership probability of each element to its cluster.

	STUD_ID	CLUS_KM_1_28_CLID	CLUS_KM_1_28_PROB
1	8 959 481	6	1.0
2	8 959 484	6	1.0
3	8 959 442	6	1.0
4	8 959 470	6	1.0
5	8 959 465	6	1.0
6	8 959 412	6	1.0
7	8 959 403	6	1.0
8	8 959 431	6	1.0
9	8 959 417	6	1.0
10	8 959 430	6	1.0
11	8 959 494	6	1.0
12	8 959 409	6	1.0
13	8 959 464	6	1.0
14	8 959 418	6	1.0
15	8 959 491	6	1.0
16	8 959 456	6	1.0
17	8 959 407	4	1.0
18	8 959 433	4	1.0
19	8 959 421	4	1.0
20	8 959 492	4	1.0
21	8 959 440	4	1.0
22	8 959 472	4	1.0
23	8 959 480	4	1.0
24	8 959 428	4	1.0
25	8 959 449	4	1.0

Fig. 3 The clustering result

- **Indexing step:** We apply the indexing technique on the table of the generated clusters. This table is not created automatically, but we created it from the clustering result by using the generated SQL code to facilitate the use of data in clusters. Therefore, we can create a B-Tree index on the table containing the data which are already classified, with Oracle.
  Example: We consider the CLUST_SECTION table which represents the table that contains the sorted data according to the section attribute. We create a B-Tree index with non unique (having duplicated values in the column) and ASC (ascending order for sorting) options, on the table containing the classified data on the section attribute:
  create non unique **index** SECTION_INDEX **on** CLUST_SECTION (SECTION);

2) *Querying phase:* In this phase, the approach must answer the user queries taking into account the data classification. It is therefore possible to reformulate these queries using the table containing the classified data and not the initial table (Student).
Example: We consider the following user query: **select** * **from** STUDENT **where** SECTION='Computer';
In this case, the query is reformulated with 'CLUST_SECTION' the representative table of classified data, and rewritten as follows:
select * **from** CLUST_SECTION **where** SECTION='Computer';
The final result will be the answer to this query.

IV. APPLICATION

We developed a query optimization tool in large databases. This tool allows users to seek satisfying answers to their queries with optimum time. We implemented our application with the Netbeans development environment and we used the Java programming language and the JavaFX technology for the graphical interfaces.

We detail afterwards the various steps of our application: After authentication, the user can select the corresponding database to get access and connect. By selecting a database, the user can visualize the contained tables and indexes in this database through buttons.

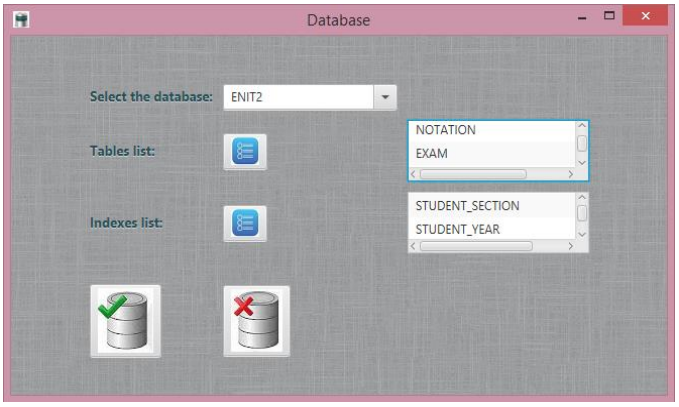

Fig. 4 Displaying the tables list and the indexes list

Across the following interface, the user can process his query according to the chosen querying mode: with or without clustering. Indeed, the user querying is an optimization across the data indexing with or without clustering. Whatever the selected mode, the user must seize his query by indicating the indexing technique and the attribute on which the indexing is carried out.

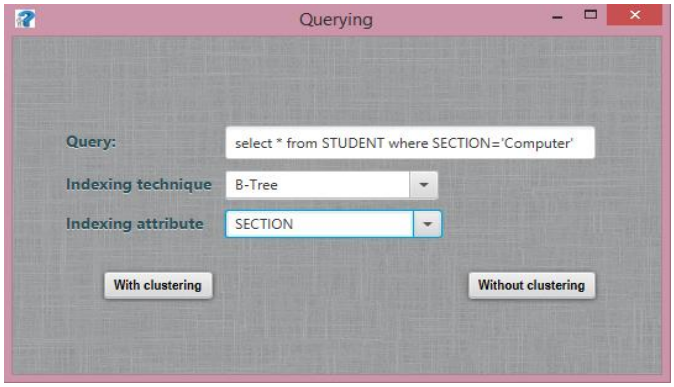

Fig. 5 Configuration of the querying parameters

The querying result depends on the selected mode. If the querying mode is without clustering, the result of the seized query is displayed in a table. In addition, the query response time will be displayed in seconds. **Example:** we seize the following query: **select** * **from** STUDENT **where** SECTION='Computer'; with the indexing technique is the B-Tree, and the indexing attribute is SECTION. The result is a table full of tuples satisfying the query conditions. Moreover, the response time is displayed in seconds.

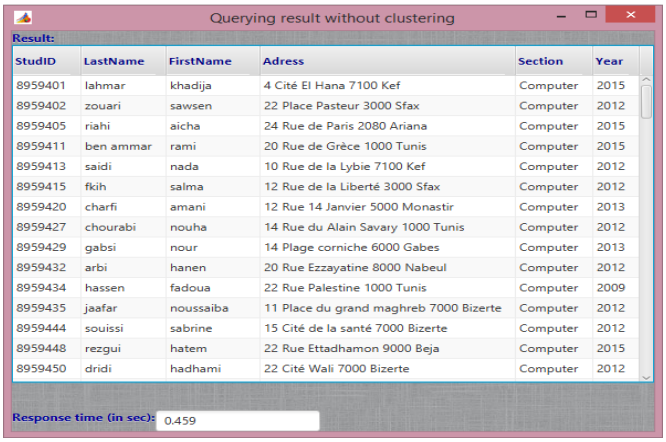

Fig. 6 Querying result without clustering

By selecting the querying mode with clustering, we obtain an interface to choose the clustering algorithm and the attribute on which the clustering is performed.

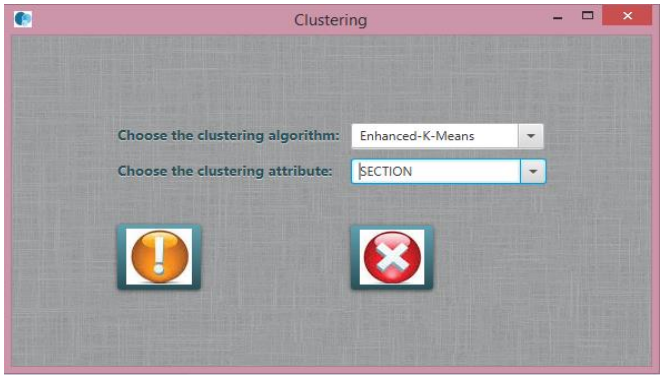

Fig. 7 Configuration of the clustering parameters

The querying result with clustering is represented by the resulting sought tuples of the query, the clusters tree and the query response time in seconds.

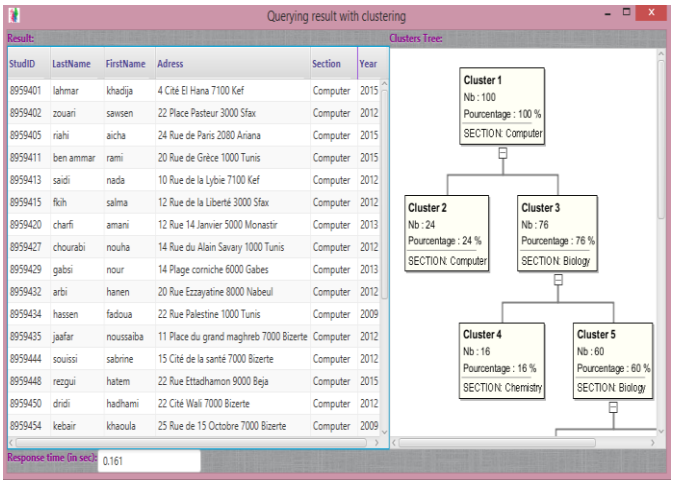

Fig. 8 Querying result with clustering

V. EVALUATION OF EXPERIMENTAL RESULTS

The general principle used to implement the queries based on the application of a clustering algorithm that is Enhanced-k-Means and the creation of a B-Tree index on the table containing the classified data. By executing a query, we calculate the response time. Then, we compare the obtained response time with that obtained by indexing without classifying the data.

Example: We consider the following query:

select * **from** CLUST_SECTION **where** SECTION='Computer';

In this example, we search for all the students in the 'Computer' section. CLUST_SECTION is the table containing the classified data. After the clustering operation, we create a B-Tree index (the default index in Oracle) on the section attribute:

create non unique **index** SECTION_INDEX **on** CLUST_SECTION (SECTION);

The comparison of the response time of the query that described above is achieved with the following query without clustering:

select * from STUDENT **where** SECTION='Computer';

Without forgetting to create the index on the section attribute:

create non unique **index** STUDENT_SECTION **on** STUDENT (SECTION);

By the same manner, we proceed for the other queries. The results are described in the figure 9 where it is clear that the response time of the queries, which are applied with our approach, decreases relative to the indexing approach without clustering. We tested these queries with a database containing 40 tuples.

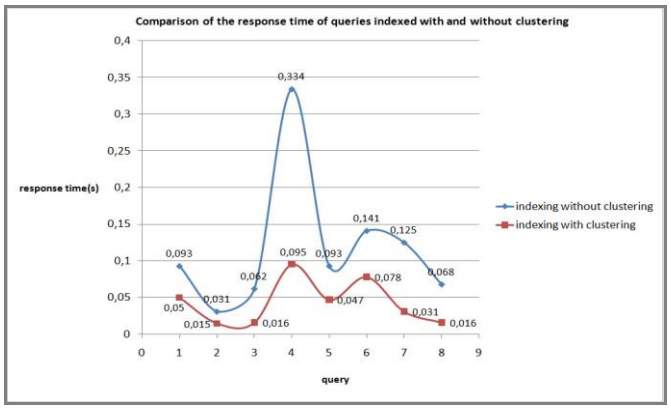

Fig. 9 Comparison of response time of queries with 40 tuples

We repeat the same process with a database containing 100 tuples to search results for the same queries made with 40 tuples. The figure 10 shows the decrease of the response time in case of data clustering before the indexing.

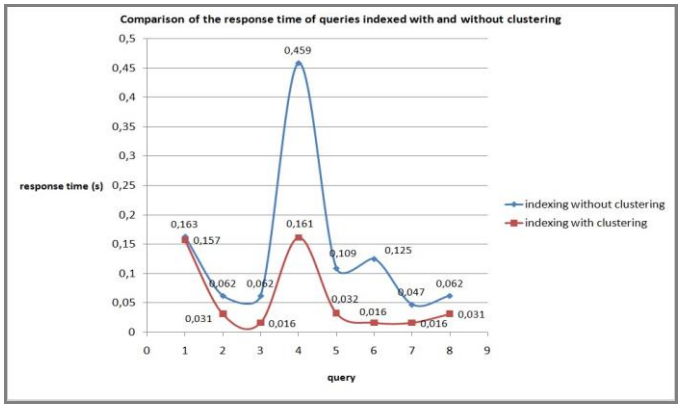

Fig. 10 Comparison of response time of queries with 100 tuples

The following figure describes the variation of the response time relative to the tuples number in case of clustering. We note that by increasing the number of tuples, it is not always evident that the response time increases and this is due to several factors such as the query complexity. Moreover, we can say that our approach remains valid by increasing the tuples number to optimize the research.

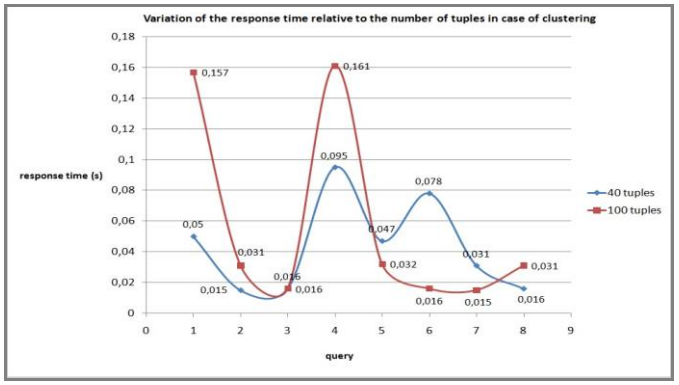

Fig. 11 Variation of the response time relative to the number of tuples

VI. CONCLUSIONS

In this paper, we proposed an approach to answer the queries for optimizing them in large databases, by using a clustering algorithm and an indexing technique. This offers a better interpretation and facilitates the queries evaluation. To realize this approach, we used the Oracle DBMS. Indeed, we represented the generated clusters by the Enhanced-k-Means clustering algorithm that is available in the Oracle Data Mining tool. Thus, we managed to interpret the clustering result to apply the B-Tree indexing technique on the classified data.

Our approach is extensible because users can choose the clustering algorithm according to their needs. In addition, it is incremental that is possible to make modifications according to the used database model. Nevertheless, our approach can be further extended by the following perspectives: 1) implement other clustering algorithms proposed by Oracle Data Mining 2) apply other algorithms implementing the fuzzy clustering 3) make additional experiments relying on other indexing techniques 4) apply our approach on flexible queries.

REFERENCES

[1] W. Kim, D. S. Reiner and D. Batory, “Query Processing in Database Systems”, Springer Science & Business Media, 2012.

[2] R. Bayer and E. McCreight, “Organization and Maintenance of Large Ordered Indexes”, pp. 245-262, Springer Berlin Heidelberg, 2002.

[3] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching”, Vol. 14, No. 2, pp. 47-57, ACM, 1984.

[4] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos and Y. Theodoridis, “R-Trees: Theory and Applications “, Springer Science & Business Media, 2010.

[5] N. Katayama and S. Satoh, “The SR-Tree: An index structure for high-dimensional nearest neighbor queries”, In ACM SIGMOD Record, Vol. 26, No. 2, pp. 369-380, 1997.

[6] J. L. Bently, “Multidimensional binary search trees used for associative searching”, Communications of the ACM, Vol. 18, No. 9, pp. 509-517, 1975.

[7] S. Berchtold, C. Böhm and H. P. Kriegal, “The pyramid-technique: towards breaking the curse of dimensionality”, In ACM SIGMOD Record, Vol. 27, No. 2, pp. 142-153, ACM, 1998.

[8] K. I. Lin, H. V. Jagadish and C. Faloutsos, “The TV-Tree: An index structure for high-dimensional data”, The International Journal on Very Large Data Bases, Vol. 3, No. 4, pp. 517-542, 1994.

[9] J. Nievergelt, H. Hinterberger and K. C. Sevcik, “The grid file: An adaptable, symmetric multikey files structure”, ACM Transactions on Database Systems, Vol. 9, No. 1, pp. 38-71, 1984.

[10] M. Freeston, “The BANG File: A new kind of grid file”, ACM SIGMOD Record, Vol. 16, No. 3, pp. 260-269, 1987.

[11] L. Kaufman and P. J. Rousseeuw, “Finding Groups in Data: An Introduction to the Cluster Analysis”, Vol. 344, John Wiley & Sons, 2009.

[12] J. C. Dunn, “A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters”, Journal of Cybernetics, Vol. 3, pp. 32-57, 1973.

[13] J. C. Bezdek, “Pattern Recognition with Fuzzy Objective Function Algorithms”, Plenum Press, New York, 1981.