Journal of Engineering and Applied Sciences 14 (19): 7234-7240, 2019
ISSN: 1816-949X

New Indexing Strategy using Machine Learning Approach for Improving Query Performance

[1]Muna Al Masawa and [2]Sultan Almotairi
[1]Faculty of Computer Science, King Abdul Aziz University, Jeddah, Saudi Arabia
[2]Department of Natural and Applied Sciences, Faculty of Community College, Majmaah University, 11952 Majmaah, Saudi Arabia, almotairi@mu.edu.sa

Abstract: The explosive growth in data volume require new technique to handle this data in data warehouse. We know that the most frequent operation over the data is queries. Many of the queries that implement in data warehouse are being very complex and also iterative. Indexed data plays very critical key role in retrieving data and searching. There are a lot of index techniques that are proposed frequently to improve the performance of query. In this study, a new model for indexing use a machine learning algorithms is proposed and evaluated in data warehouse.

Key words: Indexing, secondary indexing k-means, clustering, big data, data warehouse, efficiency, query

INTRODUCTION

Everyday, a huge amount of data continuously stored and processed by human and machines which require a more processing and retrieval requirements associated with these huge and complex data. The two trending terms in area of data processing are big data and data warehouses. Big data is commonly described by many Vs (Stantic and Pokorny, 2014): volume which data size would be at the range of Petabytes, Exabytes and more. It has assessed that the data is been doubled every 2 years and it might be more than doubling. Velocity, it is relating on both how is the data being created quickly and how is the data should be processed quickly in order to face the need for extracting helpful information. Variety, data has several format. There are structured, unstructured, semi-structured, text, media, etc. The different forms of data are creating problems for storing, data mining and analyzing. To extract this knowledge, all the types of data must be connected together. Veracity, it is relating to manage the reliability of imprecise data and if can be able to predict with noise and abnormality. It also indicates to the fact that the collected data must be meaningful to the problem this is analyzed. Volume is one of the biggest challenge in data analysis when compared with other Vs because it is more complicated to processing and retrieve the needed data in perfect time. Data warehouse is central repositories of combined and integrated data from one or more different sources. They store current and archival data in just one single place. Having access to an effective data warehouse dramatically increases your ability to make smarter decisions. In data warehouse systems the information stored in it is also has high volume and arrived to Petabytes but it is clean, integrated, fixed and time varying and is gained over many different sources (Hoffner, 2003; Vanichayobon and Gruenwald, 1999).

Indexing techniques in database systems are essential technique, specially, in relational data base that use this technique to improving query performance. In big data and data warehouse, query must be optimized because the dataset that process becoming huge. The analytical queries are increasing in very complex way. And if it wasn't optimized accurately it might take hours to execute (Eltabakh, 2017). The technologies that used today are not convenient with the big data indexing requirement, they are not completely designed to examine distributed, extended and multi-dimensional data. Converting those techniques and strategies into big data it doesn't done directly due to the unique characteristics of data and the underlying infrastructure processing the data (Gani *et al*., 2016). In the next sections of study, we will discuss these indexing techniques and introduce a new model that combines the traditional structures of indexing with machine learning and will see how the runtime will improved.

Background

Indexing: Database index is a data structure aimed at improving the time complexity and speed up queries by

Corresponding Author: Sultan Almotairi, Department of Natural and Applied Sciences, Faculty of Community College, Majmaah University, 11952 Majmaah, Saudi Arabia

basically reduced the number of records in a table that need to be searched. Indexing is the most effective tuning method but it is often neglected during development. Indexing is considered to be one of the most effective tool for reducing query execution time. It is data retriever structure that can be used to reduce the time of processed data. Also, It is one of data mining techniques that apply features extraction to find similarity measures for detecting the relationships between data (Mpinda *et al*., 2015; Paygozar and Samadi, 2016). The proposed solution for big data indexing must be covered (Kalan and Kocabas, 2016):

Speed of search: Ability to search on huge data, more than billions and trillions rows in seconds.

Multi variable queries: It must be efficient for joining the results from variable search results.

Size of index: Index size should be a small part of original data.

Granularity: It is able to produce smaller indexes when granularity can be reduced.

Parallelism: it should be easily partitioned into sections for parallel processing.

Speed of index generation: For in processing, it must be built index at the rate of data generation. In data warehouse several techniques is existing for indexing. Every technique has specific statues. Many aspects that use in determining what the right index technique is must be built on a column (Vanichayobon and Gruenwald, 1999; Jamil and Ibrahim, 2009).

Cardinality: It is how many number of the different values in a column. We need to know if the cardinality of the index column is high or low because the indexing method will only work if the cardinally is low or high. For example, bitmap index works fine with low cardinality data only.

Distribution: The column distribution is the appearance of every distinct value of the column frequently. The distribution of the column lead in determines which index method to use.

Value range: The values range of the index column leads to select suitable index technique. There are number of query types most be known before create the index, the common queries are: exact match to a specific value, range of values, join between tables, LIKE comparison, Sorted or aggregated. The worst case running time O(n) is when we search for an unindexed and unsort database that contain n key values. So, one common solution is creating indexes over the fields in data stores which they are frequently referenced by queries. Using this pattern might improve the performance of the query by allowing applications to be more quickly locate the data to make it able to retrieve it from a data store. Some of the index structures that are widely used and some are more application or query type specific.

Indexing categories: Both non-clustered and clustered indexes are different types of index structures for any database. However, The main reason to have each one of them is is to make retrieving data (query) faster.

Clustered index: A clustered index reorders the physical sort of data in a different table, so that, each table has only one clustered index. the clustered index on a table will alter the way data are stored on disk. Generally, the cluster indexes determine order of rows which will be store on disk for this reason the reading from cluster index is faster than the reading from non-cluster index, so, it cannot have more than one cluster index on one table. One of the advantage of using cluster index when the cluster groups of data it can be accessed frequently by some queries. This process will speed the retrieval because the data exists close to the one others on the disk. There is disadvantage when using cluster index when any change is made in the value of an indexed column, it will require resorting rows to maintain order which it is an effect in the performance. It is less size than non-clustered. Index rebuilding needs to drop and create new data blocks (Siddiqa *et al*., 2017).

Non-clustered index: A non-clustered index is sorted separately from actual data, so, a table may have more than one non-cluster index. This type of index does not alter or require any sort of ordering of rows in the table. Non-clustered Indexes are usually created on non-primary key columns that will not require many range queries. If a table has more than one non-clustered indexes that will not affect the order of the rows that are stored on disk like clustered indexes. Non-clustered indexes store pointer and value to the row that holds that value (Siddiqa *et al*., 2017).

Indexing types: There are two kinds of indexes.

Primary and secondary

Primary index: Primary indexes are indexes on the row of table. Primary index requires the rows in data blocks to be

ordered on the index key which it consider cluster index. It can be created on both key and non-key columns. However, primary index alters needs to keep the rows sorted and organized in a table (Sidirourgos and Kersten, 2013).

Secondary indexes: These indexes created on one or more column values which consider non-cluster index. Either the database system or your application can create and manage secondary indexes. Not all column family databases provide automatically managed secondary indexes but you can create and manage tables as secondary indexes in all column family database systems. It facilitate query answering on attributes other than primary keys. If you have a query with a where clause that uses column values that are not part of the primary key, lookup would be slow because a full row search has to be performed. Secondary indexes make it possible to service such queries efficiently. Secondary indexes are stored as extra tables and just store extra data to make it easy to find your way in the main table (Sidirourgos and Kersten, 2013).

Indexing structures

Tree based indexing: In the tree indexing strategy, retrieval of data is done in a sorted order, following branch relations of the data item. The common type of tree based strategies are B-tree. The B-tree is an organized structure following a tree, it's make the retrieval of information very easy. The B-tree has numerous nodes, the root node is the node that is on the top and the child node is the descendent and the internal node is the node who have a child, the leaf node is the node that doesn't have a child, the leaf nodes contain the pairs of key and the pointer, the pointer inside tags is pointing to correspond records in a data file. Every node have k keys, the key that point to the sub tree is the left or right pointer, the value of the keys in the left will be less than the other one on the right of tree.

Assume that if the root node will have 2 keys n and m, the child keys on the left n must be bigger than child key on the right m and the center child node keys must become between n and m. B-tree works in a similar way to the binary tree search but in a more complex manner. B-tree indexes satisfy range queries and similarity queries. The updated version from B-trees is B+tree. In B+tree there is horizontal link that connect the leaf nodes that will make the range of queries simple, this is one of the major feature of B+tree. When we set the query range condition to key1+key2, we first locate the pointer of the start row in leaf nodes that is matching to the low bound of key 1 in a query range. It could move to fetch the next record which has the same or greater key of the current recorduntil is get the upper bound of range query key2. Fetching correspond rows inside data file over the pointers of leaf nodes. The B+tree has two types either clustered type or non-clustered. The clustered B+tree will order the rows internal the data file which has the exactly the same order of the keys in leaf node. In another way, the B+tree has ordered the data file. Generally, B+tree index is built for the primary key on table. The order of rows inside the data file when non-clustered B+tree is different from the order of keys in the leaf nodes of the B+tree. To speed up the range of queries. We may build a non-clustered B+trees for none primary key of the tables such as time stamp (Gani *et al*., 2016; Vanichayobon and Gruenwald, 1999; Qin, 2016).

Bitmap indexing: Bitmap index is the index platform that target data warehouse applications. For the property which have low cardinality that have very little number of distinct value, bitmap index is built and this will speed up the queries. For example, bitmap index is built on the column that has a gender of user. Gender has two values only, male and female, so, gender column has a low cardinality. This indexes has improved the complex performance of query, it's apply Boolean operations that has low-cost such as AND, OR and NOT in the selection query that build on multiple indexes at one time. This will help in minimizing the search space before moving to the source of data.

Assume example in Table 1 in left side has the actual data of user table. the bitmap index for the gender of male and female is appearing on the last two column of the table. The sequence of the male gender in the bitmap will show if the corresponding row contain a "male" gender which will make the bit set to 1. On the other side the bit will set on 0. Each row has only one related bit in the bitmap index. The bitmap size will be huge. If we consider the cost the benefit will not worth it Jamil and Ibrahim (2009), Chen *et al*. (2015).

Hash based indexing: Hash index doesn't store the values but their hashes. Such this indexing way reducing the size andtherefore increased speed and processing of high index fields. In this case, when a query using Hash indexes will not be compared with the value of the field, but the hash value of the desired hash fields. The main purpose of hash indexes is representing high dimensional

Table 1: Bitmap Index example

User ID	User name	Gender	Birthday	Other attributes	Bitmap index for "male"	Bitmap index for "female"
5	Tom	Male	1980.01	...	1	0
6	Mary	Female	1980.02	...	0	1
7	June	Female	1979.12	...	0	1
14	Mark	Male	1979.11	...	1	0
15	Kate	Female	1985.03	...	0	1
16	John	Male	1982.02	...	1	0
7	...					

data with small binary code to gain fast search results. Hash indexing accelerates information retrieval by detecting duplicates in a large dataset. It uses a hashed key which is computed by using hash-function. Though, hashing works fine most of the time with limited data size, it tends to exhibit indexing computational when data size increases. As you can see, hash indexes are only useful for equality selections. They do not support efficient range searches. Static and dynamic hashing techniques exist. Hash indexes are performing fast approximate similarity searches for high-dimensional data (Jamil and Ibrahim, 2009).

Machine learning approaches: Machine learning algorithms have been organized into categorization, this categorization has established upon the desired outcome of the algorithm. Common algorithm types include (Sullivan, 2015; Manning *et al*., 2008).

Supervised learning: Make predictions based on a set of instances which training data includes both the input and the desired results. The common type is the classification.

Unsupervised learning: Where the points of data have no labels connected with them, the purpose of an unsupervised learning algorithm is organizing the data in some form or to describe its structure. This could group it into clusters or find other ways of looking at complex data, so that, it shows simpler or more organized. The common type is the clustering. One algorithm under sthis type that will be used in this study, it called k-means cluster (Fig. 1).

k-means clustering: k-means clustering algorithm which is one of the most straightforward unsupervised learning algorithms. k-means algorithm is a clustering method which refers to the problem of splitting a set of objects according to some problem-dependent measure of similarity. It is simple, fast in implement, easy to understand and be able to handle with large-scale data effectively. The main idea for algorithm is creating initial k centroids where, k is the parameter to the algorithm the

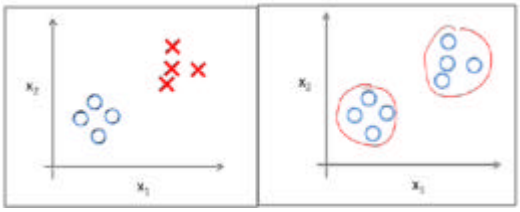

Fig. 1: Classifying vs. clustering

number of clusters. Then, every data point will be assigned to the nearest centroid which centroid of each cluster is updated based on the mean (average) of all data points in the cluster. This process finished when all points assign to appropriate cluster (Jacksi and Badiozamany, 2015; Savita and Shrivastava, 2016; Kadhim *et al*., 2016).

Literature review: Jamil and Ibrahim (2009). In this study, researchers considered three main Ideas. First of all, compare the indexing technique. Second, identify the factors that consider choosing the right index technique for data warehouse application. Then, discuss the evaluation of the indexing technique on the base of various forms of queries for data warehouses. This study will take close look on execution estimation of the three types of data warehouses queries with three indexed techniques that are different and also to detect the effect of the size of variable data with respect to the complexity of space and time.

Gani *et al*. (2016) in this study, there are 48 index techniques that had studied and compared depend on sixty articles that is discussed same topics. The performance of index techniques is analyzed based on two factors, characteristics and the requirements of indexing big data. A lot key future research topics with potential to track the progress and deployment of AI indexing. Qin (2016) in this study, researchers analyzing the structure and characteristics of Indexes that support a range of queries. In applications of big data, the data volume is so big, so, we concern about how these indexing scheme is solved the problem. We had been designing a number of experiments for comparing the indexing structures in big data situations. They analyzing the results, after that they

give several advices on appropriate situations of index. Their results are rules for database Administrator to choose distinctive indexes in physical design of a database. Jacksi and Badiozamany (2015) in this study, researchers talked about common concepts of data index and cluster methods which are based on representatives. General index theme is presented using different clustering methods and their results are compared. They studied three representative based clustering methods. One of them is outperform others.

MATERIALS AND METHODS

Since, the indexing primary key is traditional method for processing queries that fetch data based on the value of this key, this traditional approach for indexing cannot satisfy the efficiency required for queries in data warehouse with a huge volume of data. So, in this study our proposed model is attempt to improving the traditional approach by using machine learning algorithm. The methodology for this model depends on two terms. First term is secondary index and second term is machine learning approach. Primary index will be created by traditional index methods and secondary index will be created by machine learning approaches. It can create many secondary indexes as long we need it to support the various queries. In this study we focus on one of complex types of query, it is range query because it is frequently used in data warehouses and it has many troubleshoots. This type of queries will be examined and evaluated with our model. The target type of data is the numbers because the numbers need more accurately processing and deeply comparisons to find the needed records.

In first step, primary index will created on main table, once with the B-tree index and once with bitmap index. Second step is building Secondary index as new table, this table will store the center points of the values in the main table that has a huge numbers values. Theses center points is created by k-means clustering algorithm. In k-means, given n points x1, …, xn the goal is to position k centroids c1, …, ck such the distance is minimized between each point and the nearest centroid. Every centroid represents a cluster in which a cluster consisting of all points that is closest to this centroid. These centroids are used as secondary indexing for main table The method of k means clustering explained by these steps: choose the value of k which it is number of clusters, initial k guesses for the centroids. Computing the distance from every data point (x, y) to every centroid.

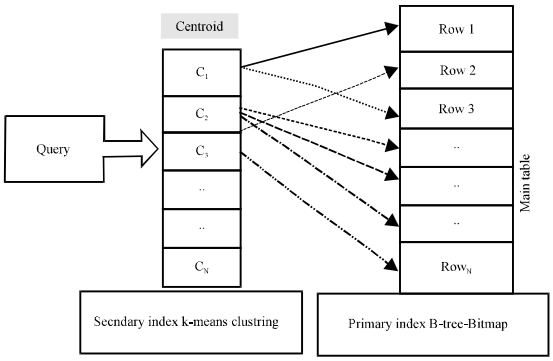

Fig. 2: The proposed index model

Then, we assign every point of data to the nearest centroid. This connection is defining the first k clusters. the distance, d between every two points (x1, y1) and (x2, y2), in two dimensions of the Cartesian plane is usually defined by the Euclidean distance measure that is showed in Eq. 1 (Fig. 2):

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

$$(x_c, y_c) = \left(\frac{\sum_{i=1}^{m} x_i}{m}, \frac{\sum_{i=1}^{m} y_i}{m} \right) \quad (2)$$

Compute the centroids, the center of points, of each new cluster from previous step. The centroid (xc, yc) of these points in k-means cluster is calculated as following in Eq. 2. Repeating the steps until the final results is reached. Third step is mapping the main table by centroids in secondary index where each group of records has same centroid. So, when search about range of values, the search will be only in specific records based on clustering centroid. Finally, queries will run on this model and results will be observed and evaluated, once when using B-tree with k-means cluster and once when using bitmap with k-means cluster.

RESULTS AND DISCUSSION

in this experiment greenplum database is used which it is a Massively Parallel Processing (MPP) database server based on open source technology Postgre SQL. Greenplum database also is including features designed to be optimized for workloads of Business Intelligence (BI). Greenplum uses the high performance architecture system for distributing the load of multi-terabyte data warehouses, also it can be using in parallel the all of a

system's resources to processing a queries (Anonymous, 2019). PostgreSQL is used in this experiments for implementing database commands. It is free and open source software. Its performance improvements are continuous with each yearly release.

So, it handles the most demand needs of the biggest insurance companies, banks and government agencies. Also it includes great performance for its unstructured data types as well latest version comes with great features which can turn it into a NoSQL database and handling big data. Range query in large database from 1000-10, 000 records is taking as case study because the range query is more complicated. k-means clustering is building using MADLib library, it is open source library using for scalable in-database analysis. It handles implementations of data parallel for mathematical, statistical and machine learning techniques on both structured data and unstructured data.

After implementing our model, the results of runtime is presented in Table 2 and Fig. 3 for B-tree and in Table 3 and Fig. 4 for bitmap. Also the size of each index is evaluated and the results is presented in Table 4. From the results, we can see there is an observed emphasis in time of processing the query and retrieve the data from large database after using the proposed model with ML approach that is introduced in this study. Using the index in general improved the query but when including ML approach, we get a better results of runtime. By comparing between two types of ML with index, we find the k-means clustering with bitmap give a better results than k-means clustering with B-tree because the centroids records that is produced by k-means satisfy the low cardinality and the bitmap work perfect in records with low cardinality. But when consider the size of indexing, the secondary indexes need extra size than primary index, since, it requires built extra table. Also the proposed indexing method performs well when there is no modification to database such as insert, update and delete as the application of OLAP. If the data is needing more changes every time, centroids start moving and every time they change position, data points have to be redistributed among them which decrease the indexing algorithms performance dramatically. The error rate on retrieved records can be enhanced by try another default values in the function of MADLib that used for creating clusters. When, we implemented the model on small table with 100 records then 1000 until 10000 records, we find the big differences between runtime values with/without including ML approach and the improvement is observed when handle the big size of database. So, we can say, this model prefer for using with big data and data warehouse.

Table 2: Run time of btree with k-means cluster

Index technique	Avg. runtime (msec)
No index	5.18
B-tree	5.676
Btree with k-means	2.526

Table 3: Run time of bitmap with k-means cluster

Index technique	Avg. runtime (msec)
No index	5.18
Bitmap	4.50
Bitmap with k-means	1.825

Table 4: Size of each indexes

Index technique	Size (k)
B-tree	181-183
Btree with k-means	155-183
Bitmap	155-183
Bitmap with k-means	172-997

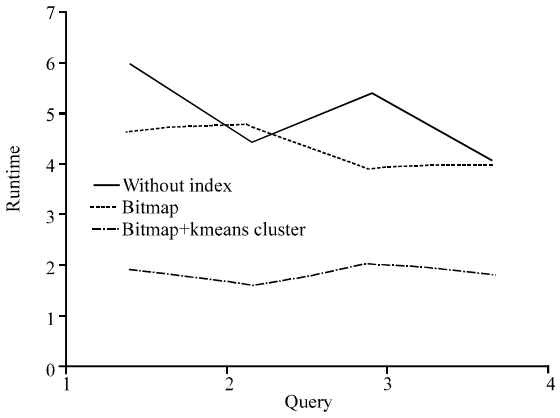

Fig. 3: Time of B-tree with k-means cluster

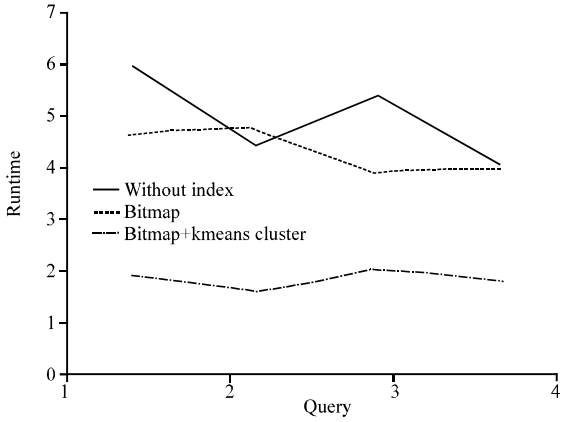

Fig. 4: Run time of Bitmap with k-means cluster

CONCLUSION

We have shown in this study how k-means clustering which it is one algorithm of machine learning approaches can be used for improving the indexing and

searching processes. Performance of these clustering techniques are compared by the traditional approach of indexing like B-tree and Bitmap. The execution time is improved and the enhancement in performance of query in general is observed. Further studies are required to extend this method to deal with database modifications. Also, we will increase the size of database and evaluate the model.

REFERENCES

Anonymous, 2019. Greenplum database administrator guide. Pivotal Software Inc., San Francisco, California, USA.https://gpdb.docs.pivotal.io/43170 / admin_guide/admin_guide.html.

Chen, Z., Y. Wen, J. Cao, W. Zheng and J. Chang *et al*., 2015. A survey of bitmap index compression algorithms for big data. Tsinghua Sci. Technol., 20: 100-115.

Eltabakh, M.Y., 2017. Data Organization and Curation in Big Data. In: Handbook of Big Data Technologies, Zomaya, A.Y. and S. Sakr (Eds.). Springer, Cham, Switzerland, ISBN:978-3-319-49339-8, pp: 143-178.

Gani, A., A. Siddiqa, S. Shamshirband and F. Hanum, 2016. A survey on indexing techniques for big data: Taxonomy and performance evaluation. Knowl. Inf. Syst., 46: 241-284.

Hoffner, V., 2003. Fundamentals of data warehouses. ACM. SIGMOD. Rec., 32: 55-56.

Jacksi, K. and S. Badiozamany, 2015. General method for data indexing using clustering methods. Intl. J. Sci. Eng. Res., 6: 641-644.

Jamil, S. and R. Ibrahim, 2009. Performance analysis of indexing techniques in data warehousing. Proceedings of the 2009 International Conference on Emerging Technologies, October 19-20, 2009, IEEE, Islamabad, Pakistan, ISBN:978-1-4244-5630-7, pp: 57-61.

Kadhim F.A., G.H.A. Majeed and R.S. Ali, 2016. Dynamic clustering for information retrieval from big data depending on compressed files. Int. J. Adv. Comput. Sci. Appl., Vol. 7. 10.14569/IJACS A.2016.070140.

Kalan, R.S. and I. Kocabas, 2016. Adaptive tools and technology in big data analytics. J. Multi. Eng. Sci. Technol., 3: 3777-3785.

Manning, C.D., P. Raghavan and H. Schutze, 2008. An Introduction to Information Retrieval. Cambridge University Press, USA., ISBN-13: 9780521865715, Pages: 482.

Mpinda, S.A.T., L.C. Ferreira, M.X. Ribeiro and M.T.P. Santos, 2015. Evaluation of graph databases performance through indexing techniques. Intl. J. Artif. Intell. Appl., 6: 87-98.

Patel, P. and D. Garg, 2012. Comparison of advance tree data structures. Intl. J. Comput. Appl., 41: 11-21.

Paygozar, H. and A. Samadi, 2016. A review of retrieval algorithms of indexing techniques on learning material. Intl. J. Comput. Sci. Inf. Secur., 14: 412-418.

Qin, X., 2016. Performance comparison of index schemes for range query of big data. Proceedings of the 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), August 13-15, 2016, IEEE, Changsha, China, ISBN:978-1-5090-4094-0, pp: 1469-1473.

Savita and S. Shrivastava, 2016. Search engine indexing using K-mean clustering technique. Intl. J. Adv. Res. Sci. Eng., 5: 218-227.

Siddiqa, A., A. Karim, T. Saba and V. Chang, 2017. On the analysis of big data indexing execution strategies. J. Intell. Fuzzy Syst., 32: 3259-3271.

Sidirourgos, L. and M. Kersten, 2013. Column imprints: A secondary index structure. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13), June 22-27, 2013, ACM, New York, USA., ISBN:978-1-4503-2037-5, pp: 893-904.

Stantic, B. and I. Pokorny, 2014. Opportunities in Big Data Management and Processing. In: Databases and Information Systems VIII, Haav, H.M.,ý A. Kalja and T. Robal (Eds.). IOS Press, Amsterdam, Netherlands, ISBN:978-1-61499-457-2, pp: 15-26.

Sullivan, D., 2015. NOSQL for Mere Mortals. Addison-Wesley, Boston, Massachusetts, USA., ISBN-13:978-0-13-402321-2, Pages: 512.

Vanichayobon, S. and L. Gruenwald, 1999. Indexing techniques for data warehouses queries. Master Thesis, The University of Oklahoma, Norman, Oklahoma.