Database optimization of News articles through Indexing

Shweta Taneja
Assistant Professor
Deptt. Of CSE
BPIT, Delhi,India

Charu Gupta
Assistant Professor
Deptt. Of CSE
BPIT, Delhi,India

Ankita Mohan Saxena,Jatin Rijhwani,Sanya Malhotra
Deptt. Of CSE
BPIT,Delhi,Indi

***Abstract*— With rapid advances in information technology, there is a plethora of information available. Out of this huge amount of data, a user (in need of information) requests the system in the form of a query. Normally the systems are not expected to return the actual information but only documents containing that information. In order to retrieve relevant information out of this huge database, the database has to be optimized in order to process the query efficiently. In this paper, indexing as a database optimization technique has been implemented on news articles. With the present work, a structured database has been created which is then optimized. A performance evaluation has been done on numerous parameters with normal execution, execution with clustered index and non-clustered index. The results show that it can significantly improve the performance which in turn will help in improving the performance of novelty mining system.**

Keywords- **Database *optimization, Preprocessing, Information retrieval, Indexing.***

I. INTRODUCTION

In today's information age, it is easy to store large amounts of data. However, although the amount of data available to us is continuously growing, our ability to gather this information and use it remains constant. Imagine the time savings if we are only presented with novel information to read, while the old or redundant information is filtered out. Thus, novelty mining [1] can help to single out novel information out of a massive set of text documents. The term novelty (derived from Latin word Novus for "new") is the quality of being new, or following from that, of being striking, original or unusual. In novelty mining, users are able to send different documents to be tested for its relevance and novelty. Due to the millions of data in the database, the insertion and selection of data have to be kept at optimum. Therefore, for that we require database optimization techniques. Database optimization is a technique to improve the query performance with indexing and statistics.

Database optimization techniques like Indexing can separate data structures that allow DBMS to locate particular records in the file of base table more quickly. Optimizing data types for a particular column in a table can reduce the CPU workload. This reduces time needed to retrieve and insert data into the database. Research of optimizing database systems has been an ongoing process where Database Management Systems (DBMS) have been evolving rapidly. Also, as lifestyles are becoming fast paced, slow and inefficient database applications are unacceptable. One of the key challenges and motivations is to pre-process the dataset. This paper includes techniques like Database Indexing that can be employed for optimization of database. A database index is a method to improve the speed of data recovery operations on a database table . It helps by reducing the time spent in database search as well as making it more efficient. The software tool used is SQL SERVER 2005, open source database software. The contributions of this paper are thus twofold. Firstly, to design and develop the optimizing techniques for SQL SERVER 2005 database for retrieval of relevant information, which has not been well-studied before, and secondly, to study the novelty mining system which involves pre-processing as its first phase followed by classification and novelty mining techniques to detect novel data from a dataset. This paper spans across the three major emerging research areas of databases that include database indexing and information retrieval by query processing, pre-processing of dataset and knowledge management.

This paper is organized as follows. In the first section, brief introduction about the motivations for the research and development of database optimization is presented. In the second section literature review of optimization as well as indexing is shown. The third section comprises of the framework that we have proposed for our entire Novelty Mining system. In sections four details about the dataset used i.e. Reuters 21578 is explained. In section five and six the experiments conducted and subsequent performance evaluation is shown. Finally, at the end of this paper we conclude and give suggestions for future work in this field.

II. RELATED WORK

The major contribution in the field of optimization and novelty mining is by Flora S. Tsai. Other authors have also contributed in this area. In [1], authors have explored the feasibility and performance of novelty mining and database optimization technique on a dataset of business blogs, with a very high accuracy. Previous researches on novelty detection have focused mainly on the task of finding novel material, given a set or stream of documents on a certain topic. Authors in [2] investigated the more difficult two part task defined by TREC 2002 novelty track that is firstly, finding the relevant sentences from the documents and then finding the novel sentences from the collection of relevant sentences. The research here shows that the former step appears to be more difficult part of the task, and the performance of novelty measures is very sensitive to the presence of non relevant sentences. In [3], authors have analyzed web logs posts for various categories of cyber security threats related to detection of cyber attacks, cyber crime and terrorism, they have used Latent Semantic models such as Latent Semantic Analysis (LSA) and Probabilistic LSA, to detect keywords from cyber security web logs. LSA is also discussed in another paper [6], where a new method for automatic indexing and retrieval is demonstrated. This approach takes the advantage of implicit higher order structure in

the association of terms with documents ("semantic structure") in order to improve the detection of relevant documents on the basis of terms found in queries. In another work [5], authors have proposed experimental results on APWSJ data set. They have shown that Document-2-Sentence framework outperforms standard document level novelty detection in terms of redundancy-precision (RP) and redundancy-recall (RR). However they have suggested that D2S shows a strong capability to detect redundant information regardless of the percentage of novelty in the document. Also, in [8] authors aim to explore the performance of redundancy and novelty mining in the business blogosphere. They have adopted the mixed metric approach which combines symmetric and asymmetric metrics.

Different researchers have contributed in the area of database optimization, but either they have focused on B–Trees or indexing techniques by LSA method. None has given attention to pre processing and optimization using indexes. In our paper, we have proposed a framework which converts unstructured data of news articles to a structured form (tables) and there after indexing is performed and performance comparison is observed. This will also form basis for our future work of novelty mining, keeping in mind the constraints and challenges in natural text.

III. PROPOSED FRAMEWORK

The framework of Novelty Mining system is shown in figure 1. It is divided into three phases:-A. Pre-processing

B. Database Optimization C. Novelty Mining. The detailed explanation of these phases is given below.

A. PRE- PROCESSING

There are various pre-processing techniques that infer or extract structured representations from raw unstructured data sources. In pre processing, there are different operations car ried out like stop word removal and word stemming. Stop word removal aims to remove stop words like 'is', 'an', 'the' etc. Word stemming is the process of reducing inflected (or sometimes derived) words to their stem, basic root form- generally a written word form. E.g. running-> run, Drinks-> drink, Mangoes-> mango

1.) ALGORITHM USED FOR WORD STEMMING

We have used a modified form of Porter Stemmer Algorithm [10]. The Porter stemming algorithm (or 'Porter stemmer') is a process for performing stemming i.e., reducing the word to its root form. It is mainly used as a part of term pre-processing, that is usually done when setting up Information Retrieval systems. The algorithm stems the data using a set of rules. There are 60 rules in 6 steps of porter stemmer algorithm. These steps are:-

1. Removes plurals of the words.
2. Turns terminal y to i when there is another vowel in the stem. Maps double suffixes to single ones, eg-'ization' , 'ational' etc
3. Deals with suffixes –full, -nests etc.
4. Takes off –ant, -ence etc.
5. Removes a final –e.

In our modified porter stemmer algorithm, we remove stop words like 'is', 'an', 'the' etc along with above suffix removal. We have used java as a programming language for implementing our algorithm. The benefit of implementing porter stemmer is to enhance search process in the large pool of data and moreover to increase the efficiency of the entire system.

B. DATABASE OPTIMIZATION

Database optimization is a technique to improve the query performance with indexing and statistics. It can be defined as the optimization of resources used to increase throughput and minimize contention, enabling the largest possible CPU workload to be processed. In our paper we have used indexing to optimize the dataset.

There are two types of indexes that have been built on the data namely clustered index and non clustered index. The two types of indexes are as explained below:

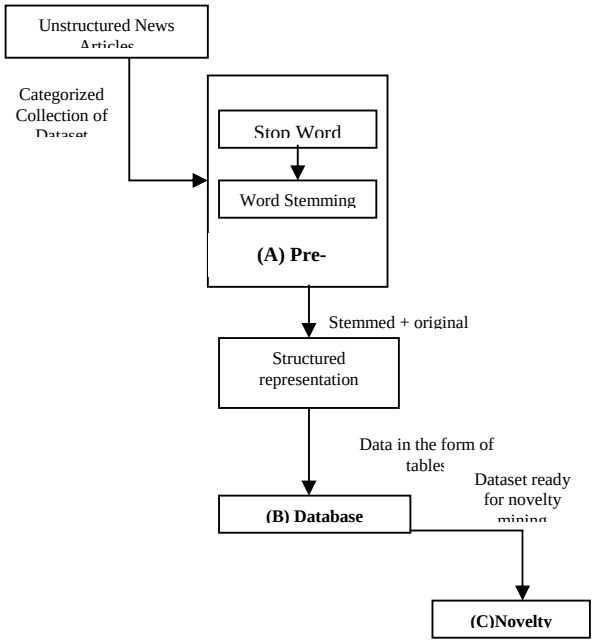


Figure 1: Proposed Framework

1) NON-CLUSTERED INDEX

The data is present in arbitrary order [12], but the logical ordering is specified by the index. The data rows may be spread throughout the table regardless of the value of the indexed column or expression. The non-clustered index tree contains the index keys in sorted order, with the leaf level of the index containing the pointer to the record (page and the row number in the data page in page-organized engines; row offset in file-organized engines). In a non-clustered index: The physical order of the rows is not the same as the index order. Typically created on non-primary key columns used in JOIN, and WHERE. There can be more than one non-clustered index on a database table. Non-Clustered indexes have structures that are different from the data rows. A non clustered index key value is used to point to data rows that contain the key value. This value is known as row locator. The structure of the row locator is determined on the basis of the type of storage of the data

pages. A heap table [11] by definition is a table that doesn't have any clustered indexes. Different pages of the heap-based table occupy different non-contiguous areas on a disk, and they are not linked together in any way.

Another case arises when there is no index defined for a table at all. In that case the address of the first IAM page of the heap table itself is stored in the sysindexes table with indid = 0 as shown in figure 3. In that respect, the abbreviation IAM is a little misleading; it would be better called as SAM (Storage Allocation Map or Space Allocation Map).

2) CLUSTERED INDEX

Clustering alters the data block [12] into a certain distinct order to match the index, resulting in the row data being stored in order. Therefore, only one clustered index can be created on a given database table. Clustered indices can greatly increase overall speed of retrieval, but usually only where the data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items is selected.

Since the physical records are in this sort order on disk, the next row item in the sequence is immediately before or after the last one, and so fewer data block reads are required. The primary feature of a clustered index is therefore the ordering of the physical data rows in accordance with the index blocks that point to them. Some databases separate the data and index blocks into separate files, others put two completely different data blocks within the same physical file(s). An object is created where the physical order of rows is the same as the index order of the rows and the bottom (leaf) level of clustered index contains the actual data rows.

C. NOVELTY MINING

Novelty mining is the identification of new or unknown information from a given set of text documents. It is useful in personal newsfeeds, information filtering, as well as many other fields where duplicate information may be returned to the users. The major components of novelty mining are Novelty scoring, Novelty decision making and Performance evaluation [2].

A novelty score is a calculated value that determines the novelty of a document and depends largely on the novelty metric that is selected. For comparison of a relevant document to its history documents following two standard forms of comparisons are used

- one-to-one comparison :

Where the current sentence is compared with each of the previous sentences, then, the maximum of the redundancy scores obtained is compared against a threshold (a) to finally decide whether the current sentence is redundant. If the maximum redundancy score exceeds a, the current sentence is detected as redundant.

- all-to-one comparison:

Where the current sentence is compared to the pool of all the previous sentences, in order to generate the redundancy score

Novelty decision setting: after obtaining the novelty score of the incoming document, the system will make a final decision on whether a document is novel or not based on the novelty decision point.

Performance evaluation: the F-score is a popular evaluation measure that is used for evaluating the results of novelty mining as well as information retrieval for measuring search, blog classification and query classification performance.

IV. DATASET

A. REUTERS 21578

We have used Reuters 21578 dataset [9] in our work. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from Reuters Ltd. (Sam Dobbins, Mike Topliss, Steve Weinstein) and Carnegie Group, Inc. (Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg) in 1987. We have used a subset of this complete dataset for our study. The dataset is divided into six categories namely companies, exchanges, organizations, people, exchanges and topics.

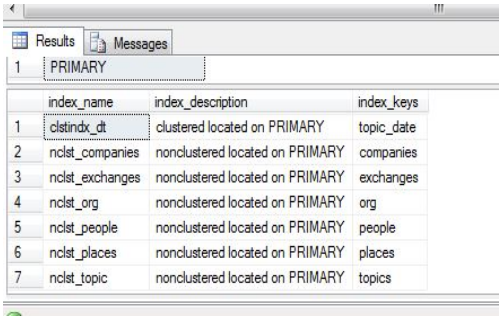
B. TOOL USED

The database software that we have used is MICROSOFT SQL SERVER 2005.Microsoft SQL Server 2005 is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). SQL Server 2005 (formerly codenamed "Yukon") was released in October 2005. It included native support for managing XML data, in addition to relational data.

V. PERFORMANCE EVALUATION

Performance evaluation is the key aspect of undertaking any research work. So we have evaluated our work and finally concluded by elaborating the efficiency of our work. But before discussing the various cases of execution, we first give an overview of the work done and then its relevant efficiency.

Following figure 2 is the list of indexes that we have created on out database:-


Results Messages

1 PRIMARY

	index_name	index_description	index_keys
1	clstindx_dt	clustered located on PRIMARY	topic_date
2	nclst_companies	nonclustered located on PRIMARY	companies
3	nclst_exchanges	nonclustered located on PRIMARY	exchanges
4	nclst_org	nonclustered located on PRIMARY	org
5	nclst_people	nonclustered located on PRIMARY	people
6	nclst_places	nonclustered located on PRIMARY	places
7	nclst_topic	nonclustered located on PRIMARY	topics

Figure 2: list of indexes on the table

As it can be seen from the above figure, we have made a total of 7 indexes on our dataset (table). One of them is a clustered index while all the others are non clustered index. The significance of using such indexes has already been discussed.

Let us now look at the various aspects of query processing in the project and thus determine optimization efficiency.

CASE 1: WHEN THERE ARE NO INDEXES ON THE TABLE.

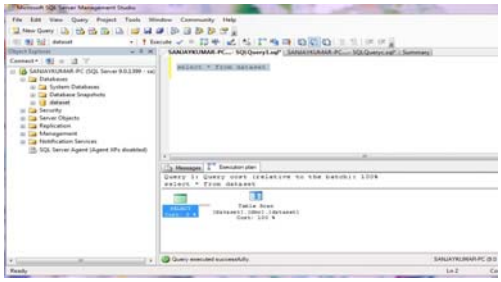


Figure 3(a): query execution plan when no indexes are present

Table Scan
Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated I/O Cost	0.0409028
Estimated CPU Cost	0.0004353
Estimated Operator Cost	0.0413381 (100%)
Estimated Subtree Cost	0.0413381
Estimated Number of Rows	253
Estimated Row Size	4221 B
Ordered	False
Node ID	0

Object
[dataset].[dbo].[dataset]
Output List
[dataset].[dbo].[dataset].topic_date, [dataset].[dbo].
[dataset].title, [dataset].[dbo].[dataset].topics,
[dataset].[dbo].[dataset].people, [dataset].[dbo].
[dataset].places, [dataset].[dbo].
[dataset].companies, [dataset].[dbo].
[dataset].exchanges, [dataset].[dbo].[dataset].org,
[dataset].[dbo].[dataset].text

Figure 3(b): query execution plan when no indexes are present

The above mentioned figures 3 and 4 give the detailed execution plan of the query being processed by the tool with various cost factors included, which can be used to evaluate the difference between the optimized and non-optimized database.

CASE 2: WHEN THERE IS NO INDEX ON THE TABLE (STILL USING WHERE CLAUSE)

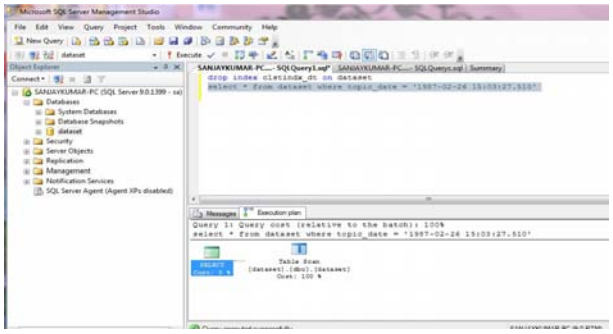


Figure 4(a): query execution plan using where clause, still no indexes on the table

Table Scan
Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated I/O Cost	0.0409028
Estimated CPU Cost	0.0004353
Estimated Operator Cost	0.0413381 (100%)
Estimated Subtree Cost	0.0413381
Estimated Number of Rows	1
Estimated Row Size	4221 B
Ordered	False
Node ID	0

Predicate
[dataset].[dbo].[dataset].[topic_date]='1987-02-26 15:03:27.510'
Object
[dataset].[dbo].[dataset]
Output List
[dataset].[dbo].[dataset].topic_date, [dataset].[dbo].
[dataset].title, [dataset].[dbo].[dataset].topics,
[dataset].[dbo].[dataset].people, [dataset].[dbo].
[dataset].places, [dataset].[dbo].
[dataset].companies, [dataset].[dbo].
[dataset].exchanges, [dataset].[dbo].[dataset].org,
[dataset].[dbo].[dataset].text

Figure 4(b): query execution plan using where clause, still no indexes on the table

CASE 3: WHEN THERE IS A CLUSTERED INDEX ON THE TABLE, BUT IT IS NOT USED.

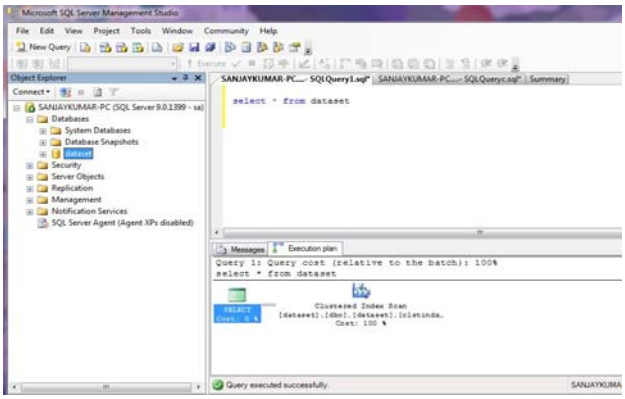


Figure 5(a): query execution plan with indexes applied, but not used

Clustered Index Scan
Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated I/O Cost	0.0409028
Estimated CPU Cost	0.0004353
Estimated Operator Cost	0.0413381 (100%)
Estimated Subtree Cost	0.0413381
Estimated Number of Rows	253
Estimated Row Size	4221 B
Ordered	False
Node ID	0

Object
[dataset].[dbo].[dataset].[clstindx_dt]
Output List
[dataset].[dbo].[dataset].topic_date, [dataset].[dbo].
[dataset].title, [dataset].[dbo].[dataset].topics,
[dataset].[dbo].[dataset].people, [dataset].[dbo].
[dataset].places, [dataset].[dbo].[dataset].companies,
[dataset].[dbo].[dataset].exchanges, [dataset].[dbo].
[dataset].org, [dataset].[dbo].[dataset].text

Figure5(b): query execution plan with indexes applied, but not used.

Here as shown by figure 5 we can see that in place of using the “table scan” here “clustered index scan” is used, which improves the query processing even though the index is still not used optimally here, as the time required to fetch the values from the table is still the same.

CASE 4: WHEN THERE IS A CLUSTERED INDEX ON THE TABLE AND IT IS USED AS WELL.

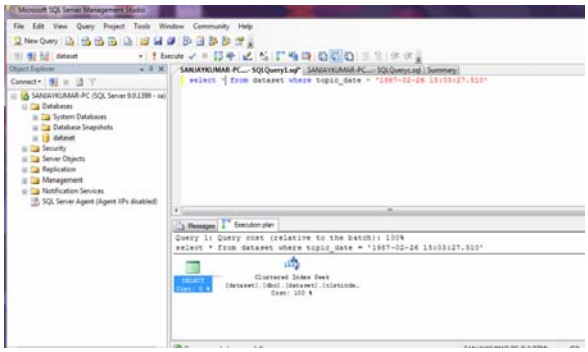


Figure 6(a): query execution plan using clustered index

Clustered Index Seek
Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Operator Cost	0.0032831 (100%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	4221 B
Ordered	True
Node ID	0

Object
[dataset].[dbo].[dataset].[clstindx_dt]
Output List
[dataset].[dbo].[dataset].topic_date, [dataset].[dbo].[dataset].title, [dataset].[dbo].[dataset].topics, [dataset].[dbo].[dataset].people, [dataset].[dbo].[dataset].places, [dataset].[dbo].[dataset].companies, [dataset].[dbo].[dataset].exchanges, [dataset].[dbo].[dataset].org, [dataset].[dbo].[dataset].text
Seek Predicates
Prefix: [dataset].[dbo].[dataset].topic_date = CONVERT_IMPLICIT(datetime,[@1],1,0)

Figure 6(b): query execution plan using clustered index

As we can now notice from the above figure 6 that in place of "table scan" and "clustered index scan", here "clustered index seek" is used. Hence there is optimal use of index present on the database and which is also reflected in the cost of overall query execution.

CASE 5: QUERY EXECUTION USING NON CLUSTERED INDEX.

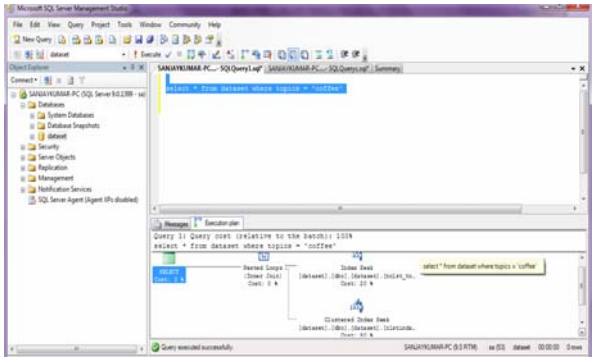


Figure 7(a): Query execution plan with non clustered indexes

Index Seek
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001625
Estimated Operator Cost	0.0032875 (20%)
Estimated Subtree Cost	0.0032875
Estimated Number of Rows	5
Estimated Row Size	26 B
Ordered	True
Node ID	1

Object
[dataset].[dbo].[dataset].[nclst_topic]
Output List
Uniq1002, [dataset].[dbo].[dataset].topic_date, [dataset].[dbo].[dataset].topics
Seek Predicates
Prefix: [dataset].[dbo].[dataset].topics = 'coffee'

Figure 7(b): Query execution plan with non clustered indexes

Clustered Index Seek
Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Operator Cost	0.0129345 (80%)
Estimated Subtree Cost	0.0129345
Estimated Number of Rows	1
Estimated Row Size	4185 B
Ordered	True
Node ID	3

Object
[dataset].[dbo].[dataset].[clstindx_dt]
Output List
[dataset].[dbo].[dataset].title, [dataset].[dbo].[dataset].people, [dataset].[dbo].[dataset].places, [dataset].[dbo].[dataset].companies, [dataset].[dbo].[dataset].exchanges, [dataset].[dbo].[dataset].org, [dataset].[dbo].[dataset].text
Seek Predicates
Prefix: [dataset].[dbo].[dataset].topic_date, Uniq1002 = [dataset].[dbo].[dataset].[topic_date], [Uniq1002]

Figure 7(c): Query execution plan with non clustered indexes

As it can be observed from figures 7 (a),(b) and (c), in the case of non clustered index, the entire query is split into two parts and thus it uses two techniques "index seek" and "clustered index seek" both in the single query and thus leading to query optimization.

VI. PERFORMANCE COMPARISON

Now the comparison between the various execution plans is given in a tabular form for easy understanding and comparison. The numerical values depicted in the table are in the form of CPU cycles required to perform the task. As can be noticed from table 1, the execution times in case of "clustered index seek" are lowest. The query execution is optimum in this case. Also the values for "index seek" and "table scan" are similar, this is due to the fact that in both the cases the indexes are not used.

TABLE1 PERFORMANCE COMPARISO N TABLE

PARAMETER	Normal execution	Normal execution on clustered attribute before indexing	Normal execution on non clustered attribute before indexing	Execution without using clustered index after indexing	Execution with clustered index	Execution with non Clustered index
Physical Operation	**Table scan**	**Table scan**	**Table scan**	**Clustered index scan**	**Clustered index seek**	**Index seek, Clustered index seek**
Logical Operation	**Table scan**	**Table scan**	**Table scan**	**Clustered index scan**	**Clustered index seek**	**Index seek, Clustered index seek**
Estimated I/O Cost	**0.0409028**	**0.0409028**	**0.0409028**	**0.0409028**	**0.003125**	**0.003125, 0.003125**
Estimated CPU Cost	**0.0004353**	**0.0004353**	**0.0004353**	**0.0004353**	**0.0001581**	**0.001625, 0.0001581**
Estimated Operator Cost	**0.0413381**	**0.0413381**	**0.0413381**	**0.0413381**	**0.0032831**	**0.0032875, 0.0129345**
Estimated Subtree Cost	**0.0413381**	**0.0413381**	**0.0413381**	**0.0413381**	**0.0032831**	**0.0032875, 0.0129345**
Estimated number of rows	**253**	**1**	**5**	**253**	**1**	**5,1**

CONCLUSION AND FUTURE WORK

In this paper, an efficient way to optimize database has been proposed with indexing technique. The proposed work uses a large dataset of news articles. The experimental results obtained in Table I show that the proposed work optimizes the database with the execution time in clustered index seek. The execution time, as can be noticed is lowest out of all other attributes. Also the value of Index seeks and Table scan are similar, as both do not consider indexes. With the proposed work the effectiveness of optimization has been studied carefully. Further investigation to the topic reveals that novelty mining with database optimization can give good results. The concept has been worked out and can be used in future with novelty mining.

ACKNOWLEDGMENT

We take this opportunity to express our sincere thanks and deep gratitude to all those who extended their wholehearted co-operation and have helped us in completing this work successfully. We express our sincere thanks to Mr. Suyash Gupta, DBA (HCL Technologies) for his encouragement and valuable suggestions.

REFERENCES

[1] A. T. Kwee, and F. S. Tsai, "Database Optimization for Novelty Mining of business blogs", Elsevier Expert Systems with Applications 38 (2011) 11040–11047.

[2] J. Allan, C.Wade, and A. Bolivar, "Retrieval and Novelty Detection at the Sentence Level", SIGIR'03, ACM 1-58113-646-3/03/0007., Toronto, Canada, July 28–August 1, 2003

[3] F. S. Tsai and K. L. Chan "Detecting Cyber Security Threats in Weblogs Using Probabilistic Models", Springer – Verlag Berlin Heidelberg, LNCS 4430, pp. 46-57, 2007.

[4] H. Cui , M.Y. Kan and T.S. Chua, "Unsupervised Learning Of Soft Patterns For Generating Definitions From Online News", ACM 1-58113-844-X/04/0005, WWW 2004, May 17–22, 2004.

[5] Flora S. Tsai · Yi Zhang, "D2S: Document-to-sentence framework for novelty detection", Received: 18 June 2009 /

Revised: 22 July 2010 / Accepted: 11 December 2010© Springer-Verlag London Limited 2010

[6] Scott Deerwester, Susan T. Dumais, Richard Harshman, "Indexing by latent semantic analysis", Journal of the American Society for Information Science (1986-1998); Sep1990; 41,6; ABI/INFORM Global Pg.391.

[7] F. S. Tsai and K. L. Chan, "Redundancy and novelty mining in the business blogosphere". The Learning Organization, Vol. 17 Iss: 6 pp. 490 – 499, Emerald Article,2010..

[8] The Text Mining Handbook Advanced Approaches in Analyzing Unstructured Data, Ronen Feldman, Bar-Ilan University, Israel James Sanger ABS Ventures, Waltham, Massachusetts.

[9] The dataset source: [http://kdd.ics.uci.edu/databases/reuters21578 [Accessed on: January 2013]

[10] Porter-Stemmer algorithm for preprocessing of dataset http://tartarus.org/martin/PorterStemmer [Accessed on: January 2013].

[11] http://msdn.microsoft.com/enus/library/aa964133(v=SQL.90).aspx [Accessed on: February 2013]

[12] http://en.m.wikipedia.org/wiki/Database_index [Accessed on: February 2013]

AUTHORS PROFILE

Shweta Taneja was born in 1981 in India. She completed her B.E. Computer Science and Engineering in 2003 and M. Tech in Computer Science and Engineering in 2009 from NSIT, Delhi in 2009. She is currently pursuing Ph.D from Delhi Technological University, Delhi. She is working as Assistant Professor in Computer Science Department at Bhagwan Parshuram Institute of Technology, Affiliated to GGSIPU, Delhi. Her areas of interest include Data Warehousing, Data Mining and Database Management Systems, Information Retreival, Text Mining.

Charu Gupta was born in 1987 in Delhi, India. She completed her B.E. in Computer Science & Engineering in 2009 and M.Tech in Computer Science and Engineering in 2011 from JSS Academy of Technical Education, Noida. She is working as Assistant Professor in Computer

Science Department at Bhagwan Parshuram Institute of Technology, Affiliated to GGSIPU, Delhi. Her areas of interest include Database Management Systems, Data Mining, Information Reterival, Fuzzy Logic, Evolutionary algorithms and applications of Evolutionary Algorithms.

Ankita Mohan Saxena was born in 1991 in New Delhi, India. She has completed her B.Tech in Computer Science and Engineering from Bhagwan Parshuram Institute of Technology, IP University, Delhi, India. Her areas of interests include Data Mining, Information Reterival and Java Programming.

Jatin Rijhwani was born in 1990 in New Delhi, India. He has completed his B.Tech in Computer Science and Engineering from Bhagwan Parshuram Institute of Technology, IP University, Delhi, India. His areas of interests include Data Mining, Information Reterival and Java Programming.

Sanya Malhotra was born in 1991 in New Delhi, India. She has completed her B.Tech in Computer Science and Engineering from Bhagwan Parshuram Institute of Technology, IP University, Delhi, India. Her areas of interests include Data Mining, Information Reterival and Java Programming.