Comparative Analysis of indexing techniques for Oracle 11g, Lucene and Terrier

[1]Hiren D. Pandya, [2]Girish H. Mulchandani, [3]Dr. Tejas P. Patalia
[1]Research Scholar, [2]Asst. Professor, [3]Head of Department
[1]Computer Engineering Department,
[1]V. V. P. Engineering College, Rajkot, India

***Abstract* - Keyword based searching requires indexing techniques and ranking model. Based on indexing, retrieval is configured and accuracy of the same can be measured by various tests. In information retrieval, Lucene and Terrier are two of the most famous libraries. Oracle 11g also has full text indexing support for multi-format data. Oracle 11g, Lucene and Terrier all have its own indexing and retrieving method based on their own algorithms. This comparative analysis is capture all the necessary details to configure index and retrieval for making any search engine. To satisfy need of multi format retrieval any of the combination is used based on performance need and system configuration which is the main gest of this study.**

***Index Terms* - Information Retrieval, Full-text Searching, Indexing, Oracle 11g, Lucene and Terrier**

I. INTRODUCTION

Information Retrieval is essential to retrieval of structured and unstructured data. With the help of Information retrieval libraries and its indexers, query parser, collection, matching model and configuration search engine can be built and performance of the same can be measured by various information retrieval terms like Precision and Recall.

Variety of storage data structures pose the problem of effective retrieval. Because indexer has to look into different location to create index. But combination of data structures will increase the overall complexity of whole search system.

Figure 1 shows a complete search system. In a complete search system set of documents are given to the Indexer which create an Index. Index contains list of documents with its Id and Indexed word. On Index Top-k algorithms were applied for effective retrieval when user enter any particular term usually known as a query term. Choice of algorithm depends on ranking model which is further responsible for link the result while display the search result based on query term. Ranking algorithm always trained by learning algorithms by giving some training data set. Effective implementation of all these components will leads to effective search engine. To enable database searching some modification is required which is able to handle database content for indexing and effective merging with the existing search system.

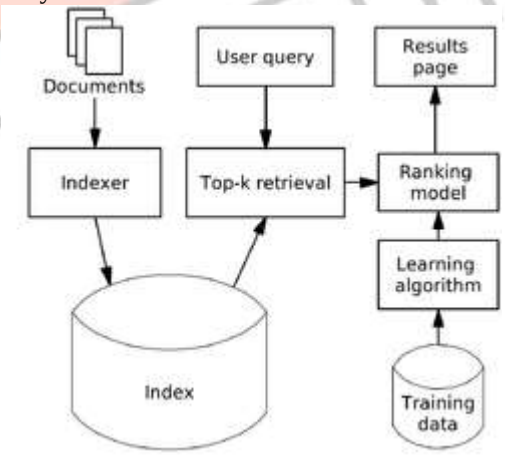

Fig 1: A Complete Search System [5]

For database indexing which type of database (i.e. Oracle, MySQL, Access etc.) is considered as a back end and need of integration is always known to the developer.

II. ORACLE INDEXING TECHNIQUE

In Oracle 11g text indexing support has been enabled by Oracle Corp. it self. By giving this functionality to the database user searching on records can be done faster. Oracle 11 g is self-sufficient to index and retrieve multi format data such as simple text files, word files, pdf files, spreadsheet files and different type of image files as well.

User can store such file in database table by specifying VARCHAR2, CLOB or BLOB data type. Even though oracle 11 g can do index on structured data and unstructured data. Because in database any of these two type data is available. Based on need, indexing type can be choose from Oracle's pre-defined indexed type as a CONTEXT, CATCAX and CTXRULE [9].

CONTEXT index type is used to implement text retrieval application which is having large collection of coherent documents [9]. CATCAX index type is used when user wants to index small fragment. It is also used when mix query performance is required [9].

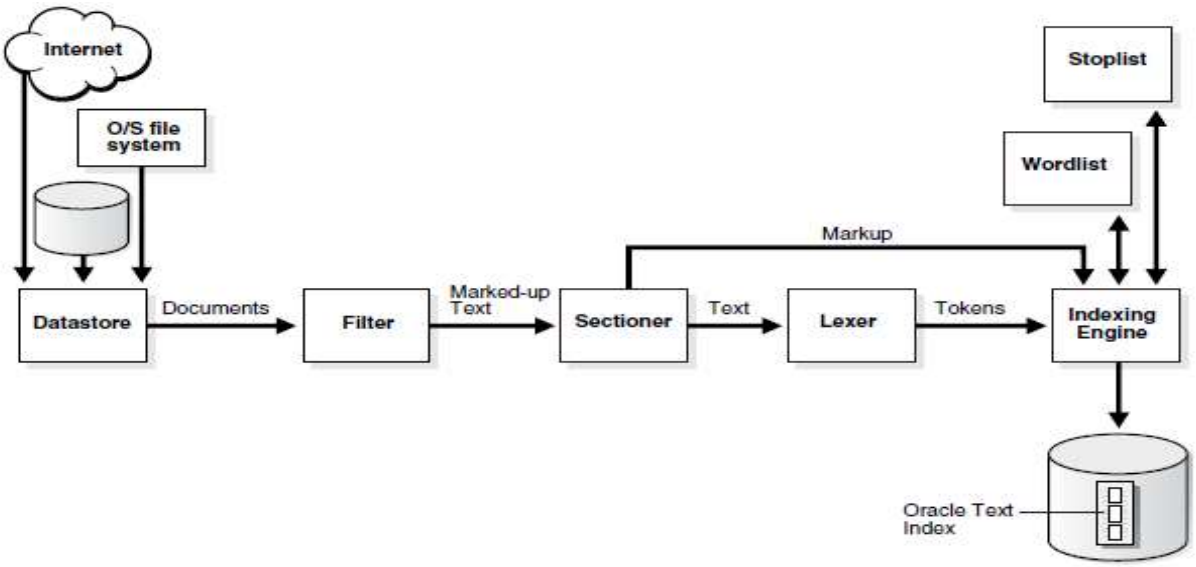

Fig 2: Oracle Text Indexing Procedure [9]

Figure 2 depicts the Oracle indexing procedure. Data comes from various sources like Internet, O/s File system and from database itself. This data goes into Datastore and it create set of documents. After this it passes through the Filters to determine document type. After being filtered, the marked-up text passes through the Sectioner that separates the stream into text and section information [9]. Sectioned data is directly passed to Indexing Engine and text data is further tokenize through Lexar. Lexar pass this tokens into Indexing Engine. Oracle Text also uses the parameters defined in your WORDLIST preference, which tell the system how to create a prefix index or substring index, if enabled and uses stoplist to bypass list of stopping words [9]. At last Indexing engine creates Index data structures which further consists of four different tables starts with prefix DR$ and ends with $I, $R, $N and $K.

$I table is represents all the tokens that have been indexed. $K table is an index organizing table which maps DOCID with ROWID. $N table is having collection of deleted DOCID and $r is used for opposite look up.

Oracle Text uses an inverse frequency algorithm based on Salton's formula. It will calculate a relevance score for a returned document in a word query [7].

$$\text{Score} = 3f\,(1+\log\,(N/n)) \qquad (1)$$

Where:

f = the number of times the term appears in the document (frequency)

N = the total number of rows in the table

n = the number of rows that contain at least one occurrence of the search term (a.k.a. document set).

After the score is calculated, the Oracle Text Engine converts it to an integer in between 1 to 100. Only oracle or any database degrades the performance of any searching mechanism [10] so combination of indexing techniques of tools is required.

III. LUCENE INDEXING TECHNIQUE

Lucene is one of the most famous information retrieval library. It is widely accepted by the industries which is written by Daug Cutting [4]. Lucene uses Apache software foundation public license. Lucene is not a complete search system but a library which can be configured on users demand. The best example of this is Nutch and ElasticSearch which uses Lucene as a back end. Its support increment indexing too. It can index over 20 MB/Min on a home machine [6]. Index size is always about 22 to 32 percent of original data. For retrieval and ranking VSM or Boolean retrieval model is used. Lucene 5.0 onwards it's also support BM25, DRF and IB (information theory based similarity) scoring model.

Figure 3 depicts the Lucene indexing process [4]. In this indexing process row content is acquired and document is built. By Indexer class it's analyzed and indexed. Through user interface user fire query and by searcher class results will render based on selected ranking model.

Figure 4 depicts the implementation directory structure of Lucene. To implement Lucene in eclipse platform it's required two main class namely IndexFiles.java and SearchFiles.java and four jar files to support parsing and configuration of retrieval.

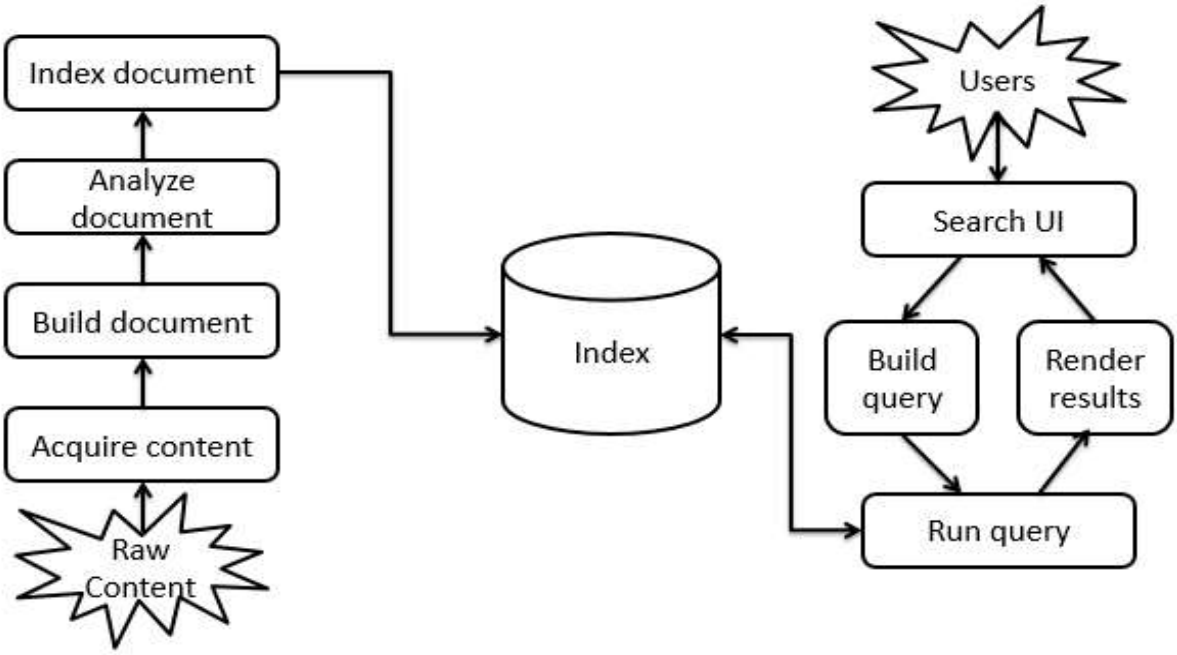

Fig 3: Lucene Indexing Technique [4]

Fig 4: Lucene Directory Structure

Lucene frequently updates it previous version for performance improvement and bug fixing. In these study lucene-5.2.1 version is being considered. Index.java class required two things mainly first is index path which specify where to create an index and doc path which specify exact location of collected document. Another SerachFiles.java class requires only index path. Because it has to perform look up when query term is fired.

In Lucene index it consist one or more segment so whenever search is perform one or more segment is refer [4]. Because of these sometime false rendering can be happen. This can be enhanced using force merging of segments. Still Lucene is one of the most stable library in all available library but it's always depends on developers how they extend its originality and up to which extent. This study is a solution of searching problem with effective retrieval configuration and also took less time for indexing with compare to database with some loss of performance.

IV. TERRIER INDEXING TECHNIQUE

Terrier is abbreviation of Terabyte Retriever and it's having state of art technique. Terrier is built by Researchers of Glasgow University, U.K. [3]. It's an open source library which comes with Mozilla Public License [3]. It's written in Java and portable. It does not support Incremental indexing. But still it's having so effective indexing techniques because of reference of more than one index in retrieval. After indexing actual data is reduce up to 5% of actual size [6]. Based on size of original data processors is required to process it. Its support distributed indexing which is essential feature of Terrier. Retrieval model of Terrier [2] are 126 DFR [1], Okapi BM25, Language modeling and TF-IDF schemas [6].

Figure 5 depicts the Terrier indexing technique [8]. Indexing starts with collection plugin that handles corpus of documents. Collection API generates stream of document objects. For each document same procedure is repeat and documents streams were generate that further passes into term pipeline. It transform series of terms that further handled by Index Builders and all indexed data will write on disk in different formats.

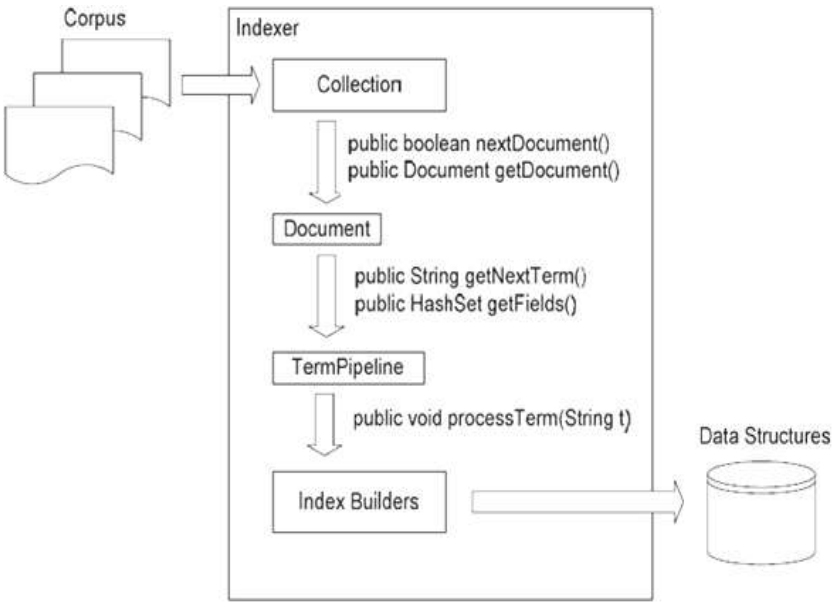

Fig 5: Terrier Indexing Process [8]

After indexing retrieval is being configured which mainly uses four index named as Direct Index, Lexicon Index, Document Index and Inverted Index. One or more index will give effective output which is put it on to higher step from other libraries and this is it's out of box functionality as well.

In Terrier, Retrieval API is used to configure retrieval by which query term is parse through various stages like pre-processing, post-processing and post-filtering [8]. After pre-processing matching can be done through matching model which uses term and document score modifiers. Further post processing is done to rank the result and display of that result as an output of post-processing and post-filtering. At the end of this one interface is needed which combines all together and works as a stand-alone application.

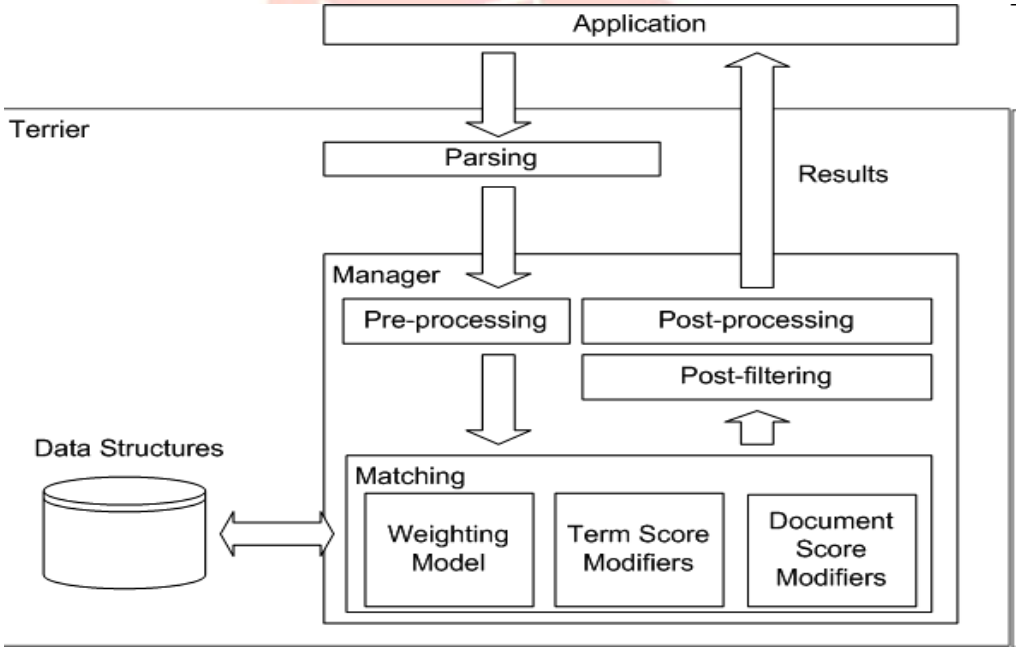

Fig 6: Terrier Indexing Process [8]

Figure 6 depicts the process of retrieval in Terrier as described above procedure. Terrie in eclipse is configured by importing terrier libraries and changes of properties which is useful to run code smoothly.

Figure 7 depicts the Terrier implementation in eclipse. In which Terrier properties were set to specify exact index location for storing of index data, its home properties for reference as well as result path to store result for further analysis.

Fig 7: Terrier Implementation Directory

As described Index and retrieval process indexing and retrieval can be perform. For verification run terrier desktop application to verify setting and for basic demo too. Terrier Desktop application gives a user interface to input your collection path. After selection of corpus press create index button for creation of index. It will iterate the posting list and traverse through various index structures and finally write all data on to the index directory. Whenever user fires query it will display the results accordingly. Figure 8 depicts the desktop application with search results.

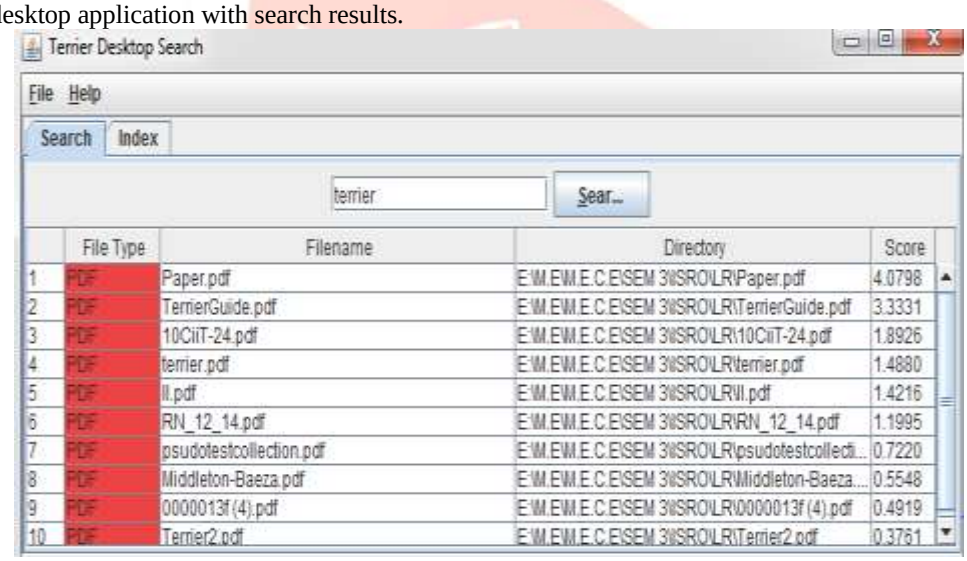

Fig 8: Terrier Desktop search

V. TEST AND ANALYSIS

For testing purpose 75 documents which are available in multi-format given to the Oracle 11g, Lucene and Terrier. All process the same documents and based on observation below table is the statistics of test results.

Table 1 Table Type Styles

Parameter	**Various tools statistics** *(for 75 various type Doc.)*		
	Oracle 11g	**Lucene**	**Terrier**
Indexing time (in second)	43	6.92	19
Searching time (in seconds)	3.77	1.83	1.2

Same thing can be depicted by graph too. Figure 9 depicts the Graph of test result.

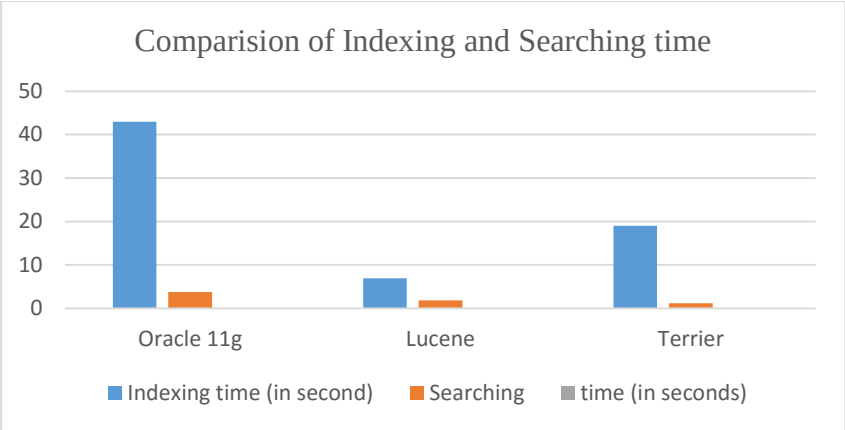

Fig 9: Graph of Test Result

VI. CONCLUSION

By analyzing tools and its techniques Oracle 11g takes more time to build index as its stores data in BLOB Format. But it's contains query took less than 5 to 10 second for searching. Lucene took very less time for indexing and retrieval but suffers from false rendering or false output. Terrier is having average time but accuracy is high with compare to Lucene and same as oracle. Only oracle is self-sufficient to retrieve images as well where as Lucene and Terrier will not retrieve any images. In contradiction Oracle takes time to build index. So, combination of Oracle and Lucene or Oracle and Terrier gives effective searching mechanism which is the gest of whole study.

REFERENCES

[1] Divergence From Randomness (DFR) Framework. Available: http://terrier.org/docs/v2.2.1/dfr_description.html. Last accessed 26 July 2010.

[2] G. Amati. Probabilistic Models for Information Retrieval based on Divergence from Randomness. PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[3] http://ir.dcs.gla.ac.uk/terrier/doc/.

[4] http://lucene.apache.org.

[5] https://manishanande.wordpress.com/.

[6] http://www.slideshare.net/KavitaGanesan/very-small-tutorial-on-terrier.

[7] http://www.oracle.com/oramag/oracle/01mar/index.html?o21int.html.

[8] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, Christina Lioma, "Terrier: A High Performance and Scalable Information Retrieval Platform" in *SIGIR Open Source Workshop* '06 Seattle, Washington, USA.

[9] Roger Ford, "Oracle Text", oracle Corporation, World Headquarters, 500 Oracle Parkway, Redwood Shores, CA 94065, USA, An Oracle Technical White paper, June 2007.

[10] Xuijin Shi, Zehnfang Wang, "An optimized full-text retrieval system based on Lucene in Oracle database," in Enterprise Systems Conference (ES), Shanghai, 2014 pp.61–65.