TECHNOCRATS INDIA COLLEGE FINDER

An internship on

SEARCH ENGINES
(Lucene and Sphinx)

Submitted by

J.JENCY MAGDALENE
2017242012
MSC(IT),4th semester

DEPARTMENT OF MATHEMATICS 2017-2022

TECHNOCRATS INDIA COLLEGE FINDER

INTERNSHIP CERTIFICATE

Certified that the intern work entitled "**SEARCH ENGINES**" is a bonafide work presented by **J.JENCY MAGDALENE** bearing 2017242012 her degree M.Sc information technology in college of engineering guindy. The report has been approved as it satisfies the companies requirements with respect to intern work by the student.

Trainer

Name: JAYARAJ.S
Designation: Senior Software Developer

C.T.O

Name: JEBAMONEY MATTHIAS
Designation: Chief Technology Officer

Table of Contents

LUCENE SEARCH ENGINE

OBJECTIVE

On the first day of my internship a task was assigned for me learning about different search engines such as lucene, sphinx, google and yahoo.

WHAT IS LUCENE?

On the upcoming days I learned about what is meant by a search engine and an overview about lucene search engine .In lucene search engine I learned that it creates short keywords for the data which is called as the indexes and when we need to search the data it searches from the generated indexes and not from the original data.

How lucene works

The process of this search engine is that first the input data is passed into a query parser in which the data is split and is passed into the analyser. The job of the analyser is to analyse the data that is it removes the noun, verbs such as an, a, is, .. etc from the search data. Then it creates an index for the data and the index writer is a class that writes and updates the indexes in the indexer class. Once the index is created we can search the data. When the search data is given it searches the data in the generated index and displays the output .Next I learned how to implement lucene search engine using java. During this implementation the package for lucene should be downloaded and imported into the working space.

Source code

Lucenetester.java

```
package javaapplication14;

import java.io.IOException;

import java.util.Scanner;

import org.apache.lucene.document.Document;

import org.apache.lucene.queryParser.ParseException;

import org.apache.lucene.search.ScoreDoc;

import org.apache.lucene.search.TopDocs;


public class Lucenetester {

   String indexdir="C:\\lucene\\Index";

   String datadir="C:\\lucene\\Data";
```

```
   Searcher s;
   public static void main(String[] args) throws IOException, ParseException
   {String name;
   Lucenetester t=new Lucenetester();
   t.createIndex();
   System.out.println("enter the data to be searched :");
   Scanner s=new Scanner(System.in);
   name=s.nextLine();
   t.search(name);
 }

private void createIndex() throws IOException{
    Indexer i=new Indexer(indexdir);
    int numind;
    long starttime=System.currentTimeMillis();
    numind=i.createIndex(datadir,new Textfilefilter());
    long endtime=System.currentTimeMillis();
    i.close();
    System.out.println(numind+" File indexed, time taken: "+(endtime-     starttime)+" ms");
   }

private void search(String searchquery)throws IOException,ParseException {
    s=new Searcher(indexdir);
    long startime=System.currentTimeMillis();
    TopDocs hit=s.search(searchquery);
    long endtime=System.currentTimeMillis();
```

```
    System.out.println(hit.totalHits +" document(s) found. Time :"+
    (endtime - startime));
    for(ScoreDoc scoreDoc : hit.scoreDocs) {
    Document doc = s.getDocument(scoreDoc);
    System.out.println("File: "+ doc.get(Luceneconstants.fpath));
    }
     s.close();
   }
}
```

Indexer.java

```
package javaapplication14;
import java.io.*;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

public class Indexer {
   private IndexWriter writer;
    public Indexer(String indexDirectoryPath) throws IOException
    Directory indexdir=FSDirectory.open(new File(indexDirectoryPath));
    writer=newIndexWriter(indexdir,new)
```

```
    StandardAnalyzer(Version.LUCENE_36),true,
    IndexWriter.MaxFieldLength.UNLIMITED
 }

public void close() throws CorruptIndexException,IOException{
   writer.close();
   }

private Document getDocument(File file) throws IOException{
    Document doc=new Document();
    Field confield=new Field(Luceneconstants.con,new FileReader(file));
    Field fnfield = new Field(Luceneconstants.fname,
    file.getName(),Field.Store.YES,Field.Index.NOT_ANALYZED);
    Field fpfield = new Field(Luceneconstants.fpath,
    file.getCanonicalPath(),Field.Store.YES,Field.Index.NOT_ANALYZED);
    doc.add(confield);
    doc.add(fnfield);
      return doc;
   }

 private void indexFile(File file) throws IOException {
   System.out.println("Indexing "+ file.getCanonicalPath());
   Document doc=getDocument(file);
    writer.addDocument(doc);}

public int createIndex(String dataDirPath,FileFilter filter)throws IOException{
    File[] files = new File(dataDirPath).listFiles();
```

```
    for (File file : files) {
    if(!file.isDirectory()
          && !file.isHidden()
          && file.exists()
          && file.canRead()
          && filter.accept(file)
         {indexFile(file);
       }
     }
     return writer.numDocs();
  }
}
```

Searcher.java

```
package javaapplication14;

import java.io.*;

import org.apache.lucene.analysis.standard.StandardAnalyzer;

import org.apache.lucene.document.Document;

import org.apache.lucene.index.CorruptIndexException;

import org.apache.lucene.queryParser.ParseException;

import org.apache.lucene.queryParser.QueryParser;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.ScoreDoc;

import org.apache.lucene.search.TopDocs;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.FSDirectory;
```

```
import org.apache.lucene.util.Version;


public class Searcher {

   IndexSearcher isearch;

   QueryParser qparser;

   Query q;


public Searcher (String indexDirectoryPath) throws IOException{

     Directory idir=FSDirectory.open(new File(indexDirectoryPath));

     isearch = new IndexSearcher(idir);

     qparser = new  QueryParser(Version.LUCENE_36,Luceneconstants.con,new StandardAnalyzer
(Version.LUCENE_36));

   }

public TopDocs search(String searchquery) throw IOException ParseException{

     q=qparser.parse(searchquery);

    return isearch.search(q, Luceneconstants.max_search);

   }


public Document getDocument(ScoreDoc sdoc) throws CorruptIndexException,IOException {

   return isearch.doc(sdoc.doc);

   }


public void close() throws IOException {

       isearch.close();

   }

}
```

Source files

Luceneconstants.java

```
package javaapplication14;

public class Luceneconstants {

  public static final String con = "contents";

  public static final String fname = "filename";

  public static final  String fpath = "filepath";

  public static final int max_search = 10;

 }
```

Textfilefilter.java

```
package javaapplication14;

import java.io.File;

import java.io.FileFilter;

public  class Textfilefilter implements FileFilter {

   @Override

  public boolean accept(File pathname) {

  return pathname.getName().toLowerCase().endsWith(".txt");

  }

   }
```

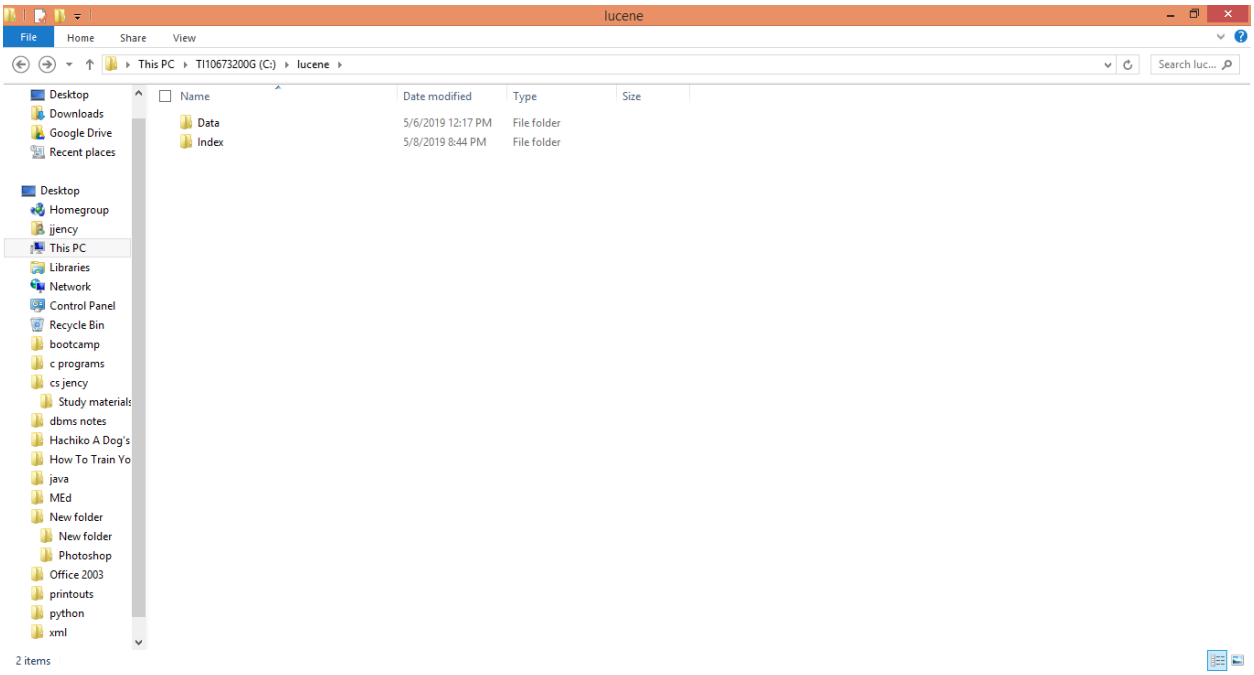


Figure 1:location of data and index directory

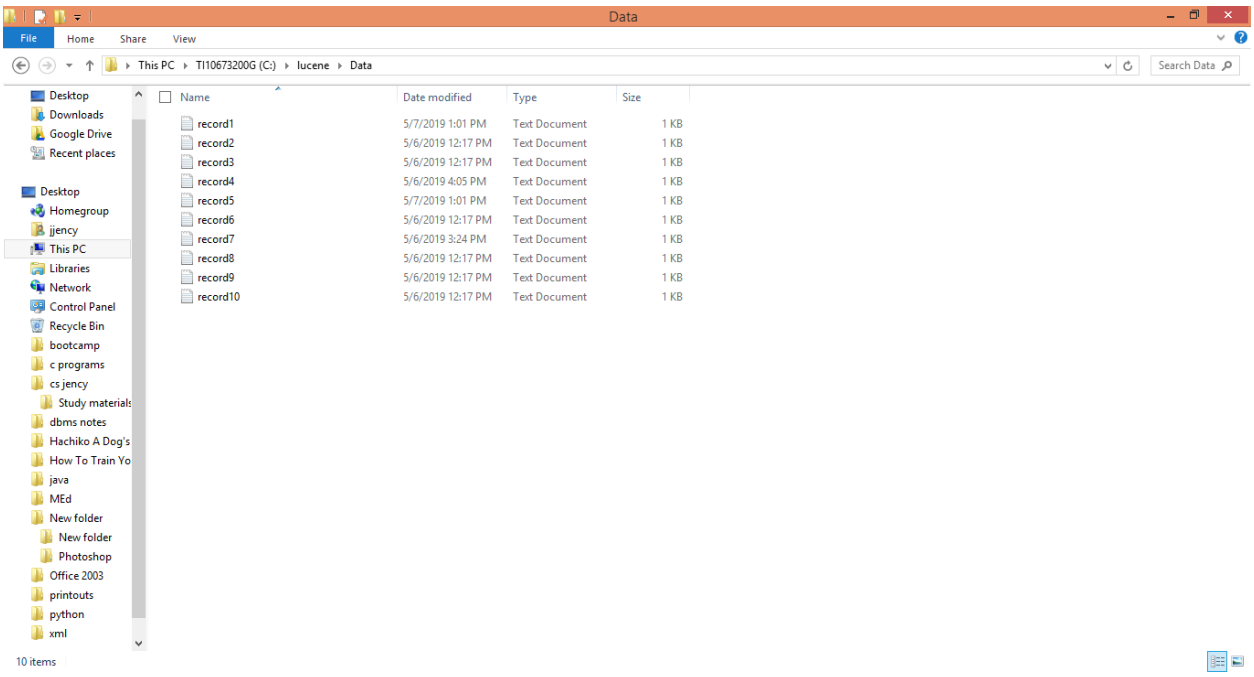


Figure 2:data to be indexed

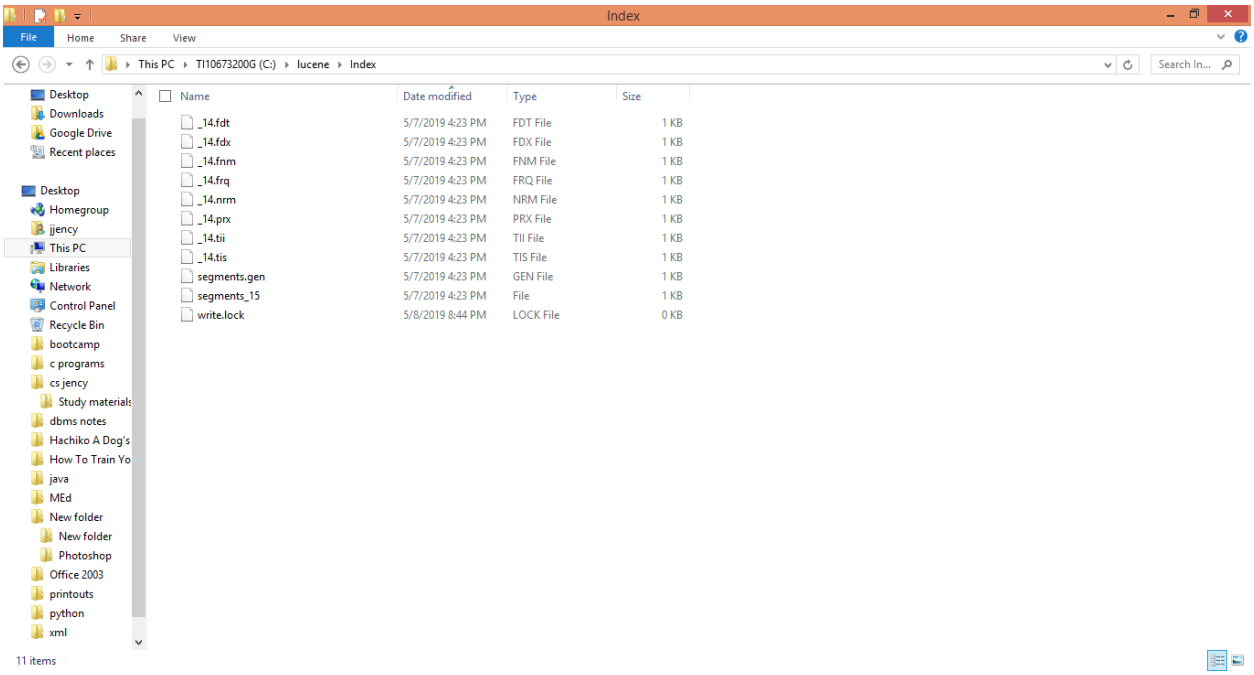


Figure 3:indexes

Output

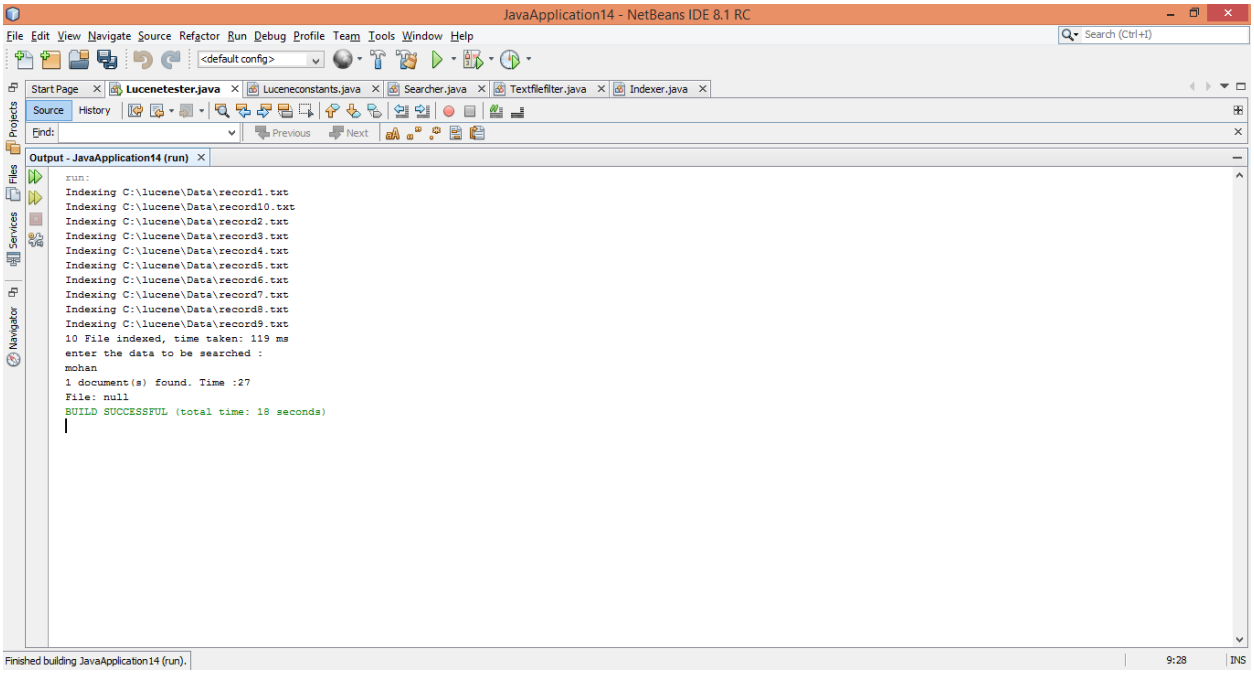


Figure 4:lucene search engine

Java database connectivity

I successfully executed the search engine and the mistake that I made was that the data to be searched was given in the program and not manually so I corrected the mistake by getting my search data from the user using scanner class. In this search engine we searched the data from a given file so we tried to search data in a database but due to the unacceptability of package (jdbc – odbc bridge) in this java version I was unable to complete that task

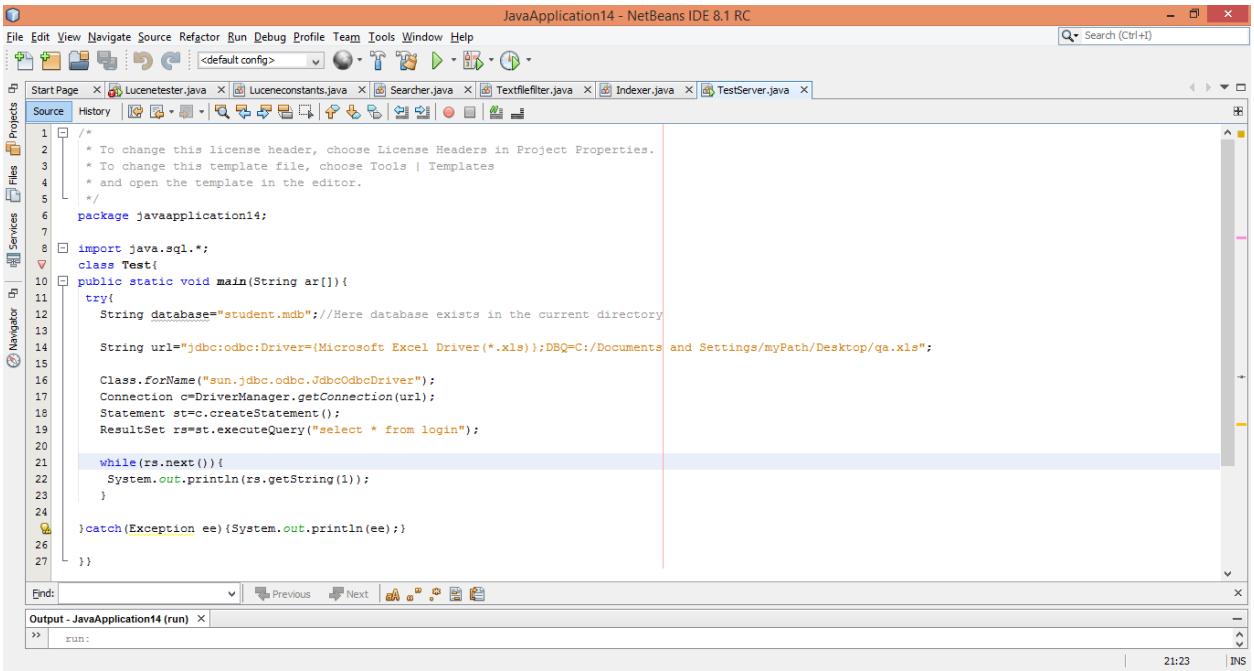


Figure 5:program for jdbc-odbc connectivity

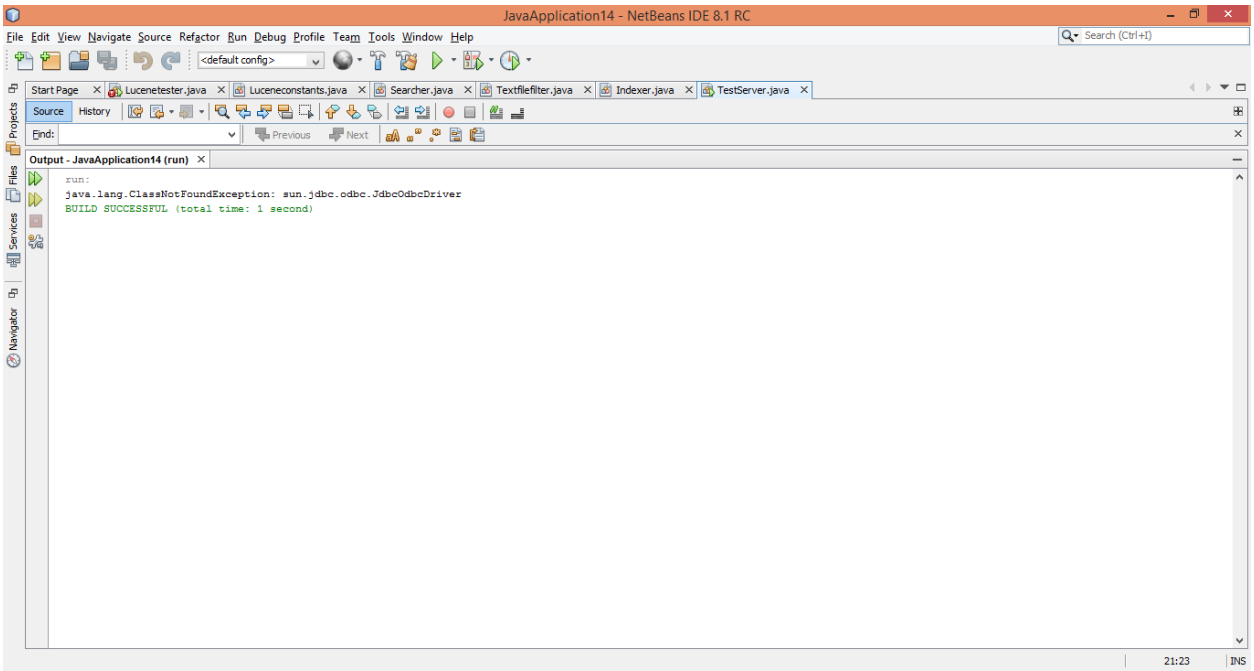


Figure 6:error

SPHINX SEARCH ENGINE

Sphnix is a search engine library which supports real time indexing. It also supports database indexing for information retrieval, which is not directly supported by lucene. Sphinx also generates an index and it searches the data in that index which is pretty much similar to lucene but it is superfast and the relevance ranking is default in sphinx but lucene on the other hand has many useful features related to web information retrieval.

Sphinx indexes

Sphinx indexes are semi-structured collections of documents. They may seem closer to SQL tables than to Mongo collections, but in their core, they really are neither. The primary, foundational data structure here is a *full-text index*. It is a special structure that lets us respond very quickly to a query like "give me the (internal) identifiers of all the documents that mention This or That keyword"

What is a config file?

In sphinx installation various versions are available, in certain versions search facility is not available by default so we need to setup that search manually. I installed the version where both indexer and search applications are available .So the important thing in sphinx is the config file where the configuration is done .The config file is a collection of different classes such as source, indexer, searchd and many more which are not that necessary for a basic search.

TASK 1

To create a plain index for a given database and search through the indexes using sphinx

Source code(config file)

Plain indexes

```
source src1
{
        type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = test
        sql_port                = 3306  # optional, default is 3306


        sql_query               = \
                SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
                FROM documents
```

```
        sql_attr_uint           = group_id
        sql_attr_timestamp      = date_added

        sql_query_info          = SELECT * FROM documents WHERE id=$id
}



index test1
{
        source                  = src1
        path                    = C:/sphinx2/data/test1
        docinfo                 = extern
        charset_type            = sbcs
}



index testrt
{
        type                    = rt
        rt_mem_limit            = 32M

        path                    = C:/sphinx2/data/testrt
        charset_type            = utf-8

        rt_field        = title
        rt_field        = content
        rt_attr_uint            = gid
}


indexer
{
        mem_limit               = 32M
}


searchd
{
        listen                  = 9312
        listen                  = 9306:mysql41
        log                     = C:/sphinx2/log/searchd.log
        query_log               = C:/sphinx2/log/query.log
        read_timeout            = 5
        max_children            = 30
```

```
        pid_file            = C:/sphinx2/log/searchd.pid
        max_matches         = 1000
        seamless_rotate     = 1
        preopen_indexes     = 1
        unlink_old          = 1
        workers             = threads # for RT to work
        binlog_path         = C:/sphinx2/data
}
```

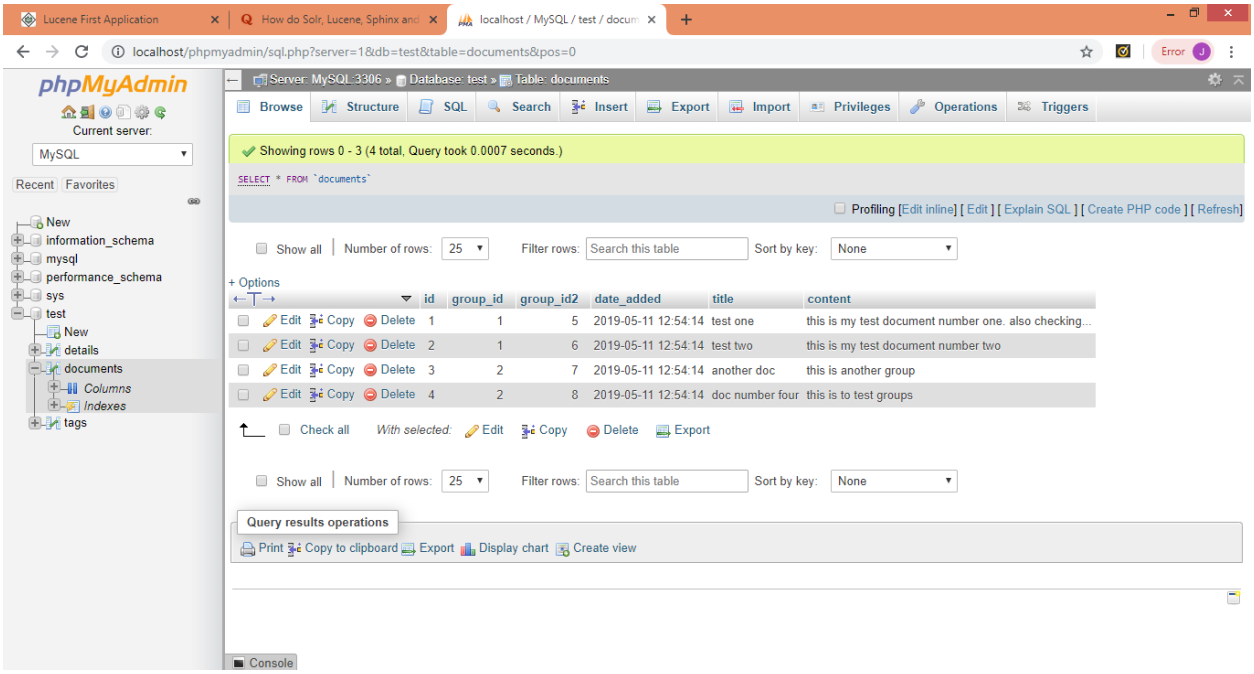


Figure 7:database to be indexed

OUTPUT

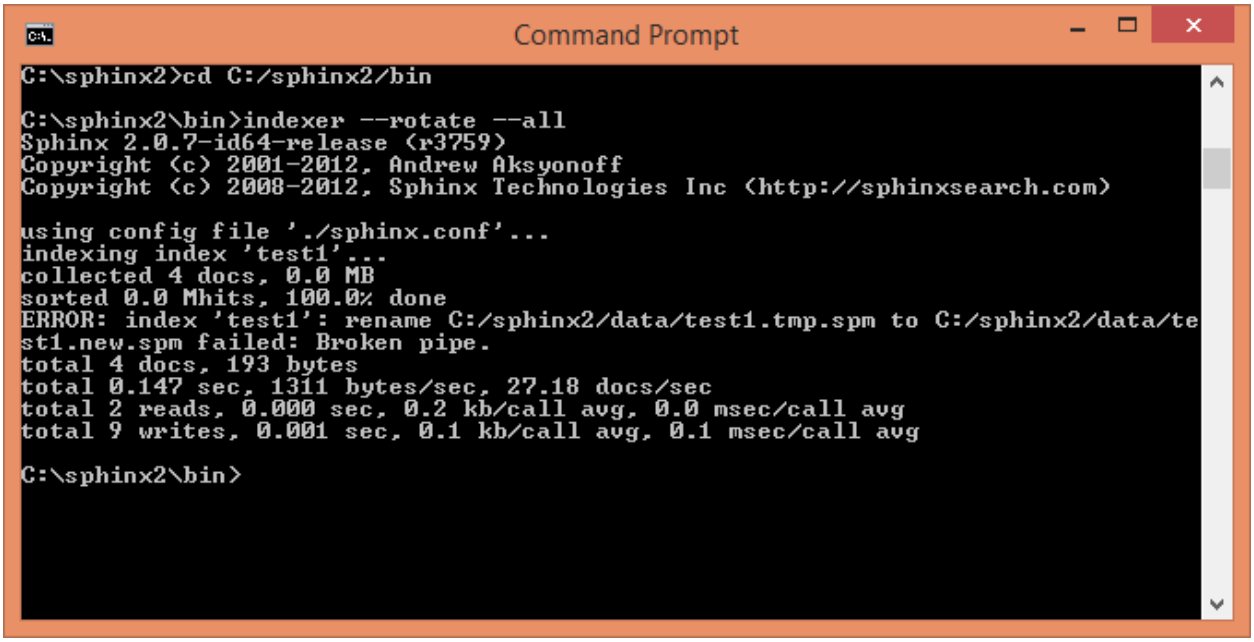


Figure 8:indexer run successfully

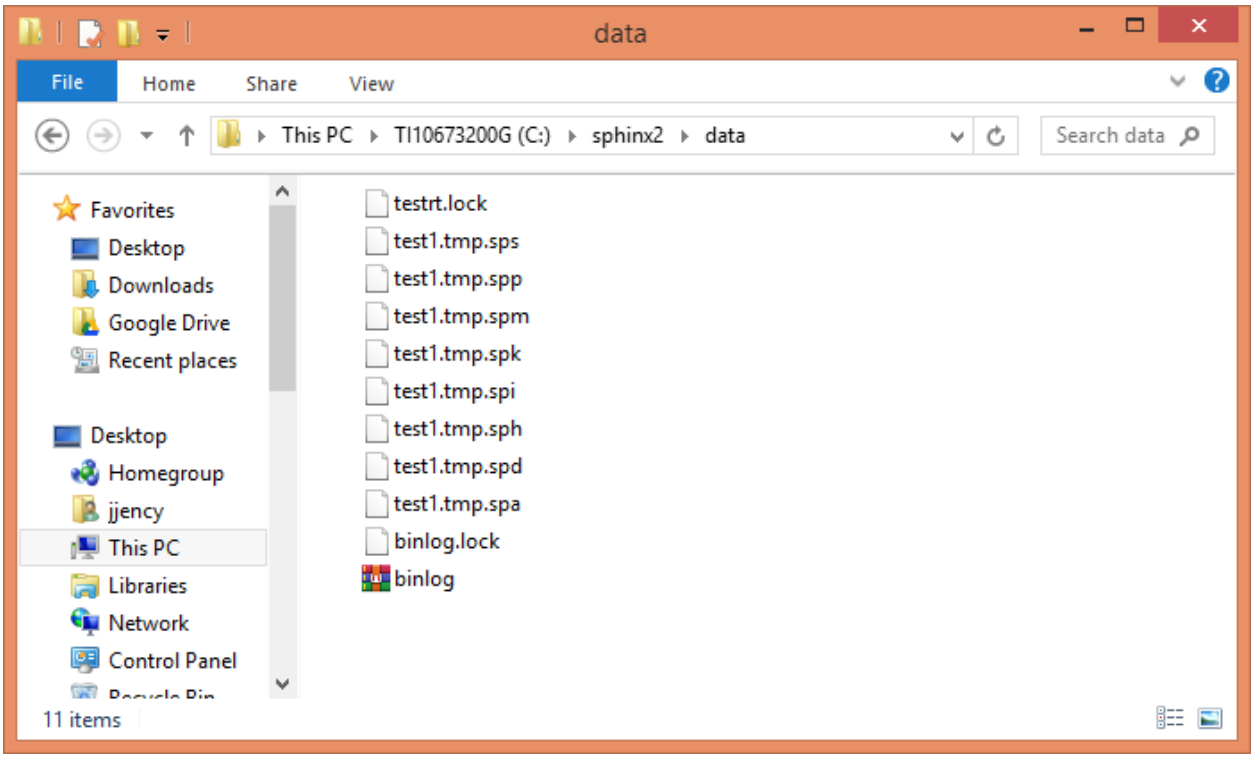


Figure 9:indexes

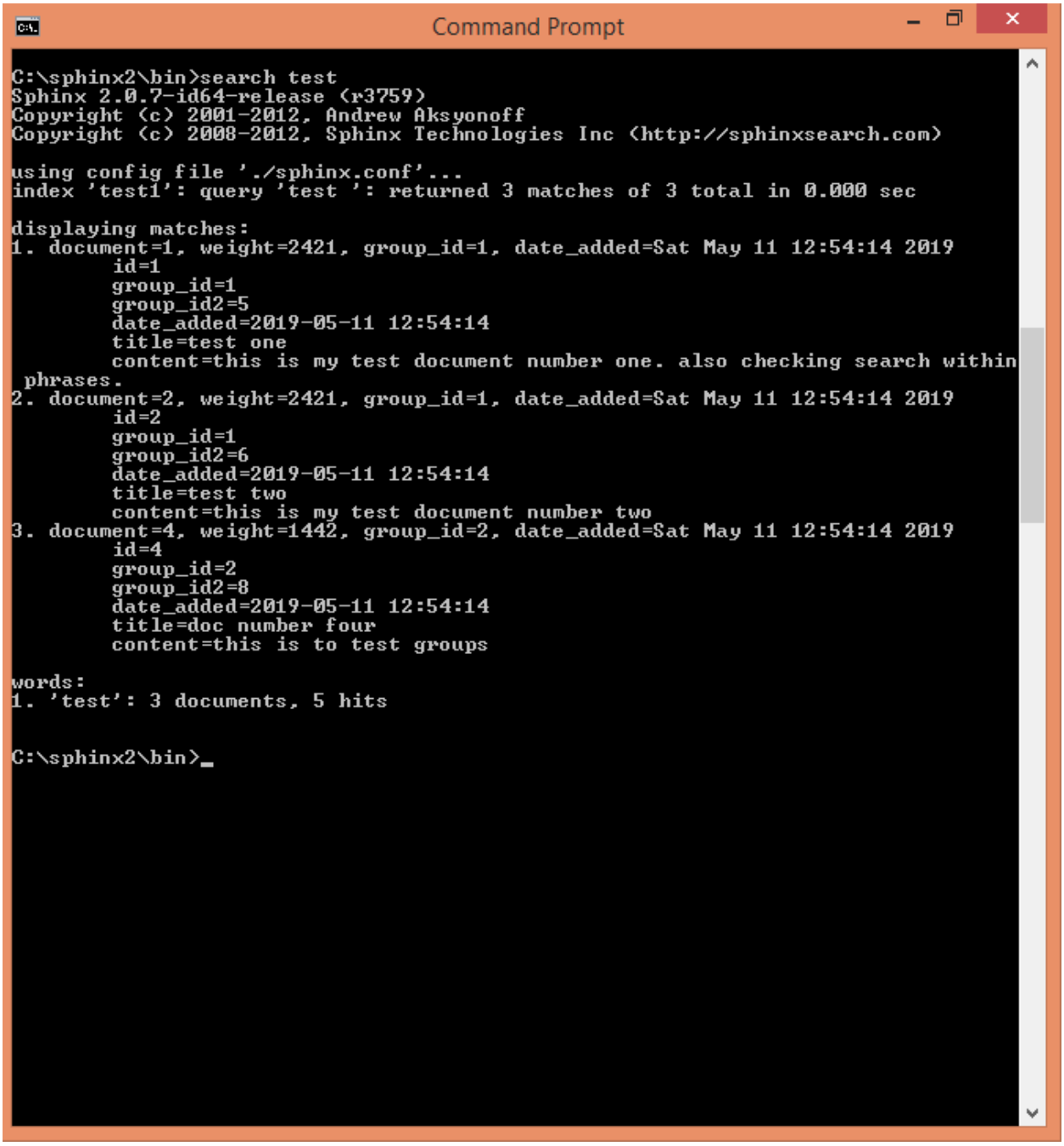


Figure 10:searching

TASK 2

To generate multiple indexes for many different tables in a database at a same time and search the data from the multiple indexes

Multiple indexes

The source class consists of the data about the database that it needs to search like the database name , password ,table name and the query .In indexer class it consists of the file path where the index to be stored. Next is the searchd class this class is used to create and start a service to run this application. If the service is not started then there is an error in the config file .we use CMD to run this application first the correct path to the directory(case sensitive) should be given otherwise there will be an error. If our config file is correct then the index will be generated in the designated path. Next job is to search the data from the generated index .So in this data is searched in a single index next we searched data using multiple indexes .

```
source src1
{
	type                    = mysql

	sql_host                = localhost
	sql_user                = root
	sql_pass                =
	sql_db                  = test
	sql_port                = 3306  # optional, default is 3306

	sql_query               = \
		SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
		FROM documents

	sql_attr_uint           = group_id
	sql_attr_timestamp      = date_added

	sql_query_info          = SELECT * FROM documents WHERE id=$id
}
```

```
source src1p0
{
   type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = test
        sql_port                = 3306  # optional, default is 3306

        sql_query               = \
                SELECT id, url, description \
                FROM details
sql_query_info          = SELECT * FROM details WHERE id=$id

}
index test1
{   type        = plain
        source                  = src1
        path                    = C:/sphinx2/data/test1.new
        docinfo                 = extern
        charset_type    = sbcs
}
index idx1
{
        type = plain
        source                  = src1p0
        path                    = C:/sphinx2/data/idx1.new.new
        docinfo                 = extern
        min_prefix_len          = 3
        charset_type    = sbcs

}
indexer
{
        mem_limit               = 32M
}

searchd
{
   dist_threads = 3
        listen                  = 9312
```

```
        listen                  = 9306:mysql41
        log                     = C:/sphinx2/log/searchd.log
        query_log               = C:/sphinx2/log/query.log
        read_timeout            = 5
        max_children            = 30
        pid_file        = C:/sphinx2/log/searchd.pid
        max_matches             = 1000
        seamless_rotate                 = 1
        preopen_indexes                 = 1
        unlink_old              = 1
        workers                         = threads # for RT to work
        binlog_path             = C:/sphinx2/data
}
```

Databases for multiple indexes

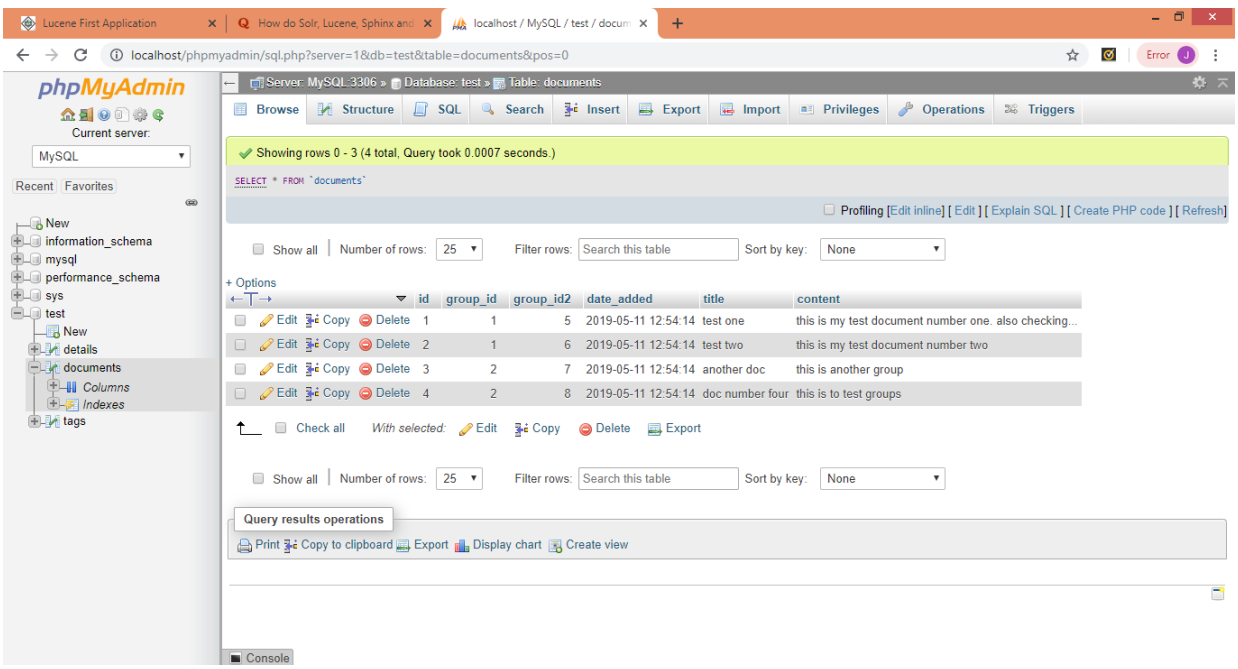


Figure 11:database 1

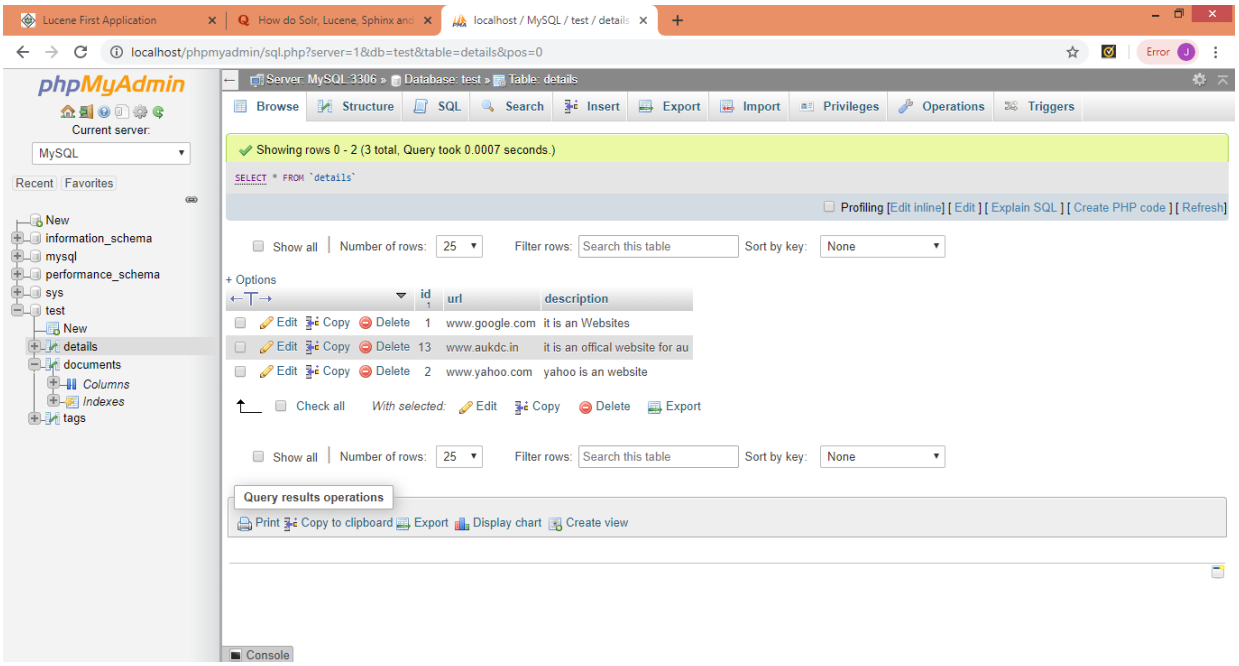


Figure 12:database 2

OUTPUT

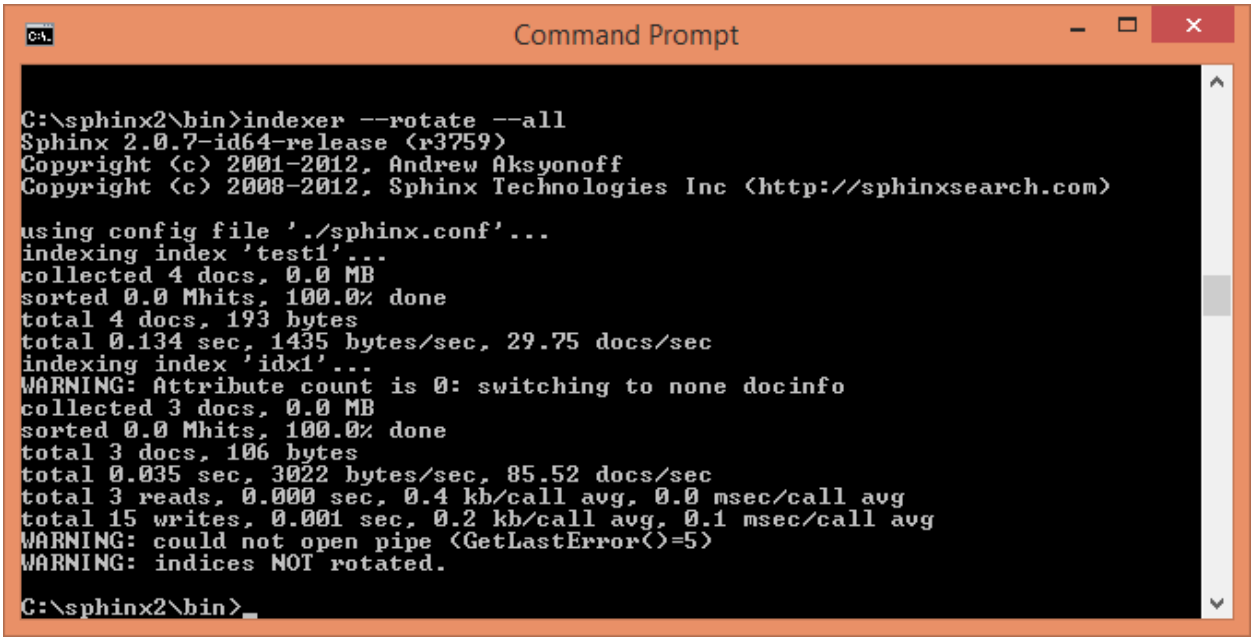


Figure 13:indexer run sucessfully

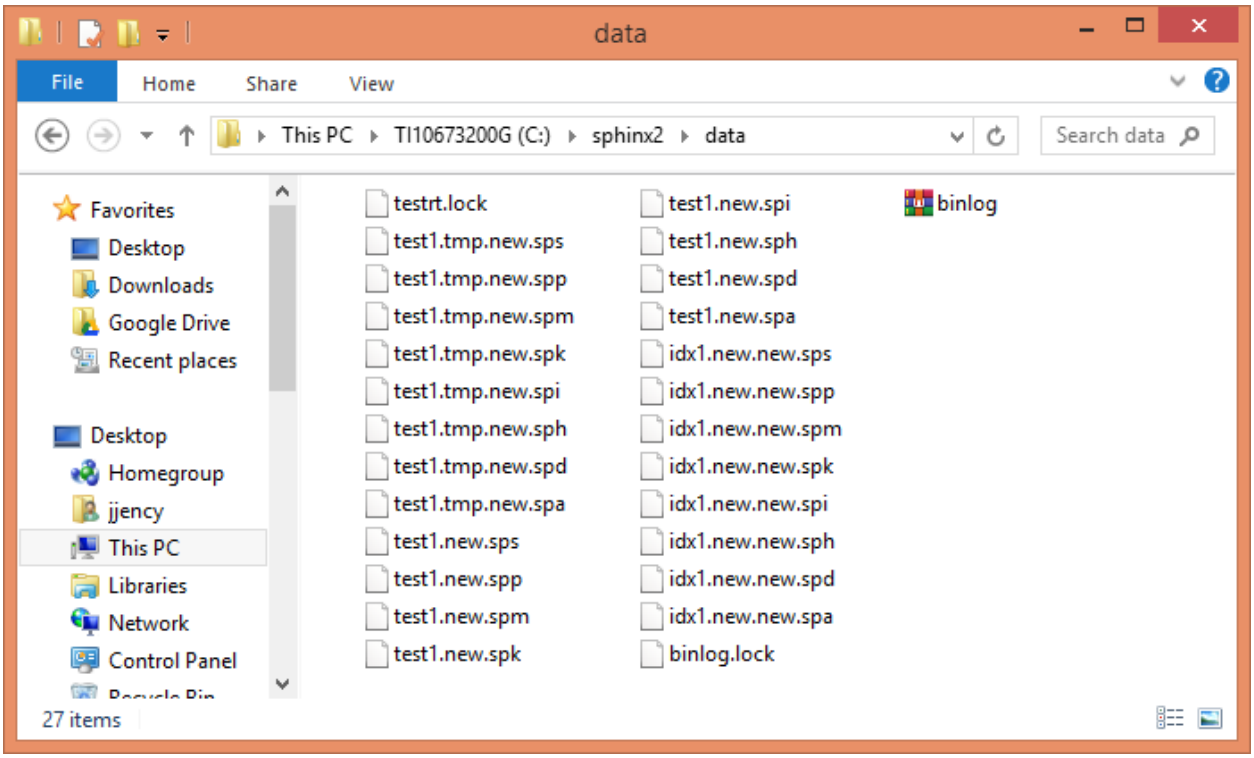


Figure 14:indexes

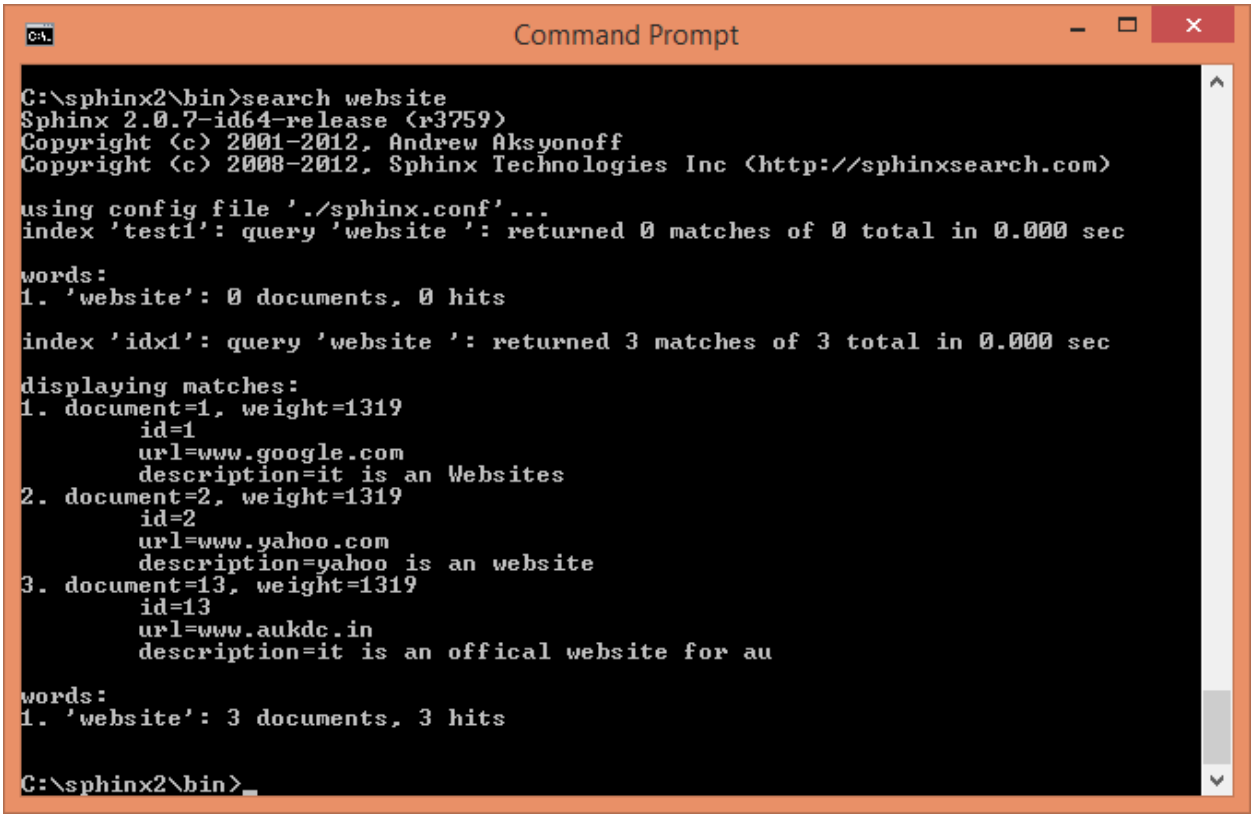


Figure 15:searching data in multiple indexes

TASK 3

To generate indexes for two different databases and search the data from the indexes

Searching through two different databases

```
source src1
{
        type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = test
        sql_port                = 3306  # optional, default is 3306

        sql_query               = \
                SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
                FROM documents

        sql_attr_uint           = group_id
        sql_attr_timestamp      = date_added

        sql_query_info          = SELECT * FROM documents WHERE id=$id
}


source src1p0
{
   type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = table
        sql_port                = 3306  # optional, default is 3306

        sql_query               = \
                SELECT id, name, mark \
                FROM students
sql_query_info          = SELECT * FROM students WHERE id=$id

}
index test1
```

```
{   type          = plain
            source                      = src1
            path                        = C:/sphinx2/data/test1.new
            docinfo                     = extern
            charset_type    = sbcs
}
index idx1
{
            type = plain
            source                      = src1p0
            path                        = C:/sphinx2/data/idx1.new.new
            docinfo                     = extern
            min_prefix_len              = 3
            charset_type     = sbcs

}
indexer
{
            mem_limit                   = 32M
}

searchd
{
   dist_threads = 3
            listen                      = 9312
            listen                      = 9306:mysql41
            log                         = C:/sphinx2/log/searchd.log
            query_log                   = C:/sphinx2/log/query.log
            read_timeout                = 5
            max_children                = 30
            pid_file            = C:/sphinx2/log/searchd.pid
            max_matches                 = 1000
            seamless_rotate                     = 1
            preopen_indexes                     = 1
            unlink_old                  = 1
            workers                             = threads # for RT to work
            binlog_path                 = C:/sphinx2/data
}
```

databases locations

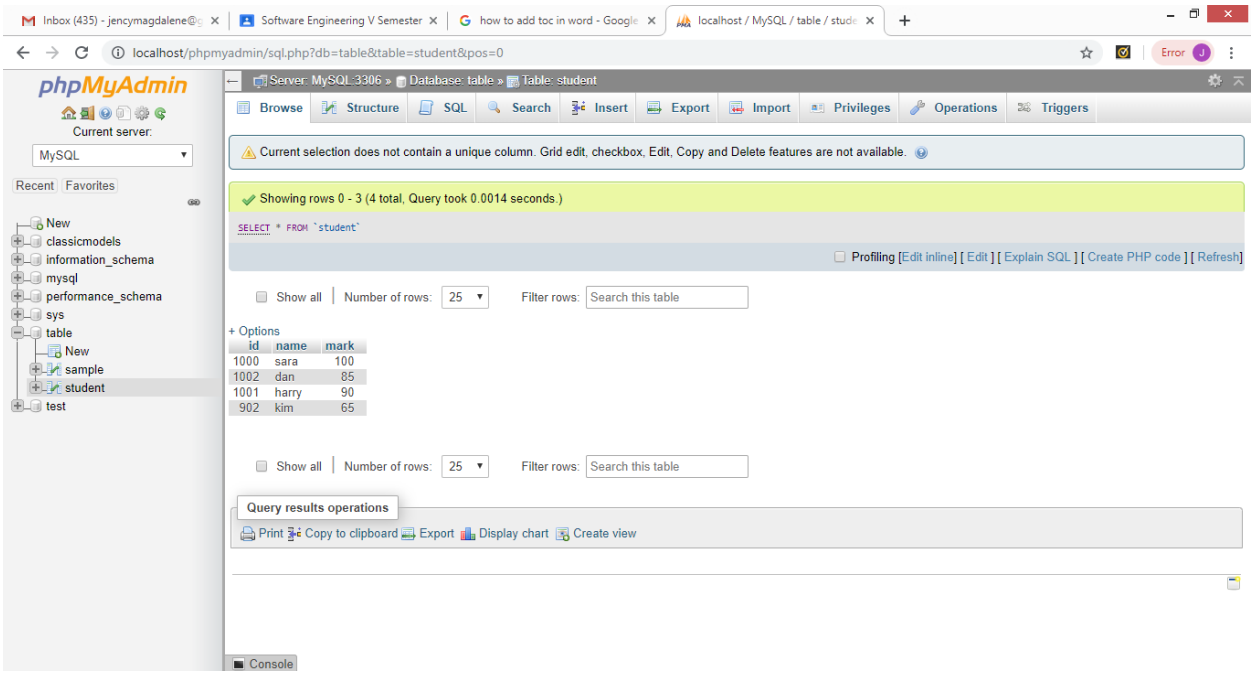


Figure 16:student table

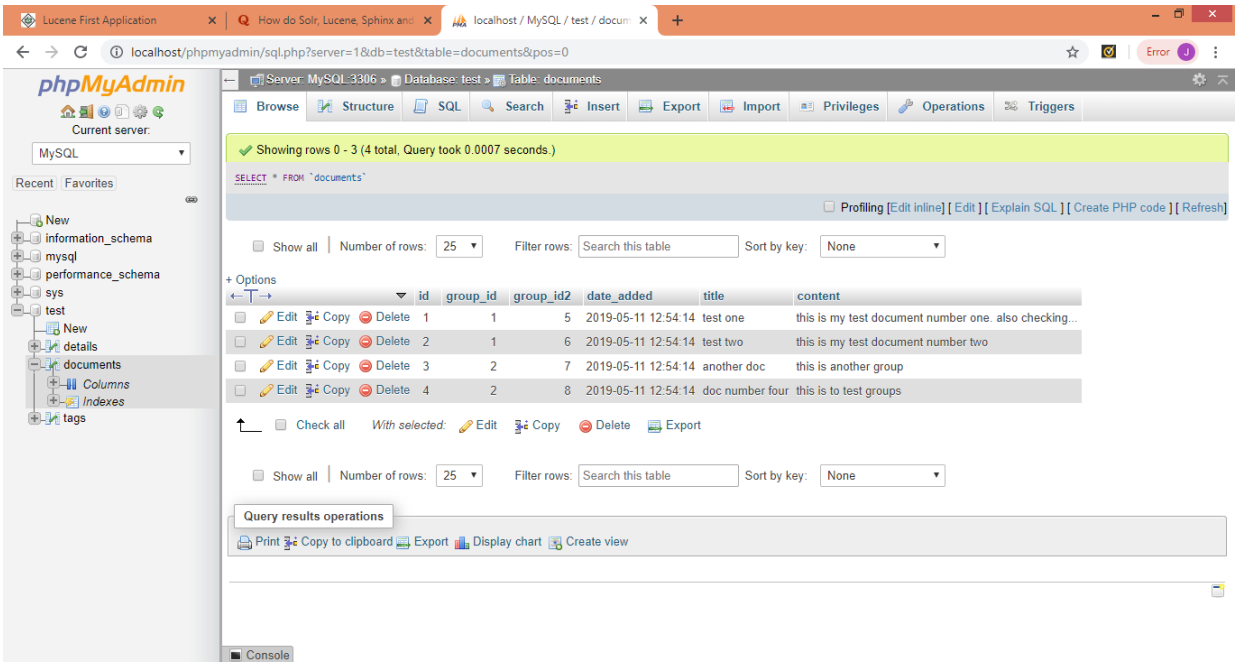


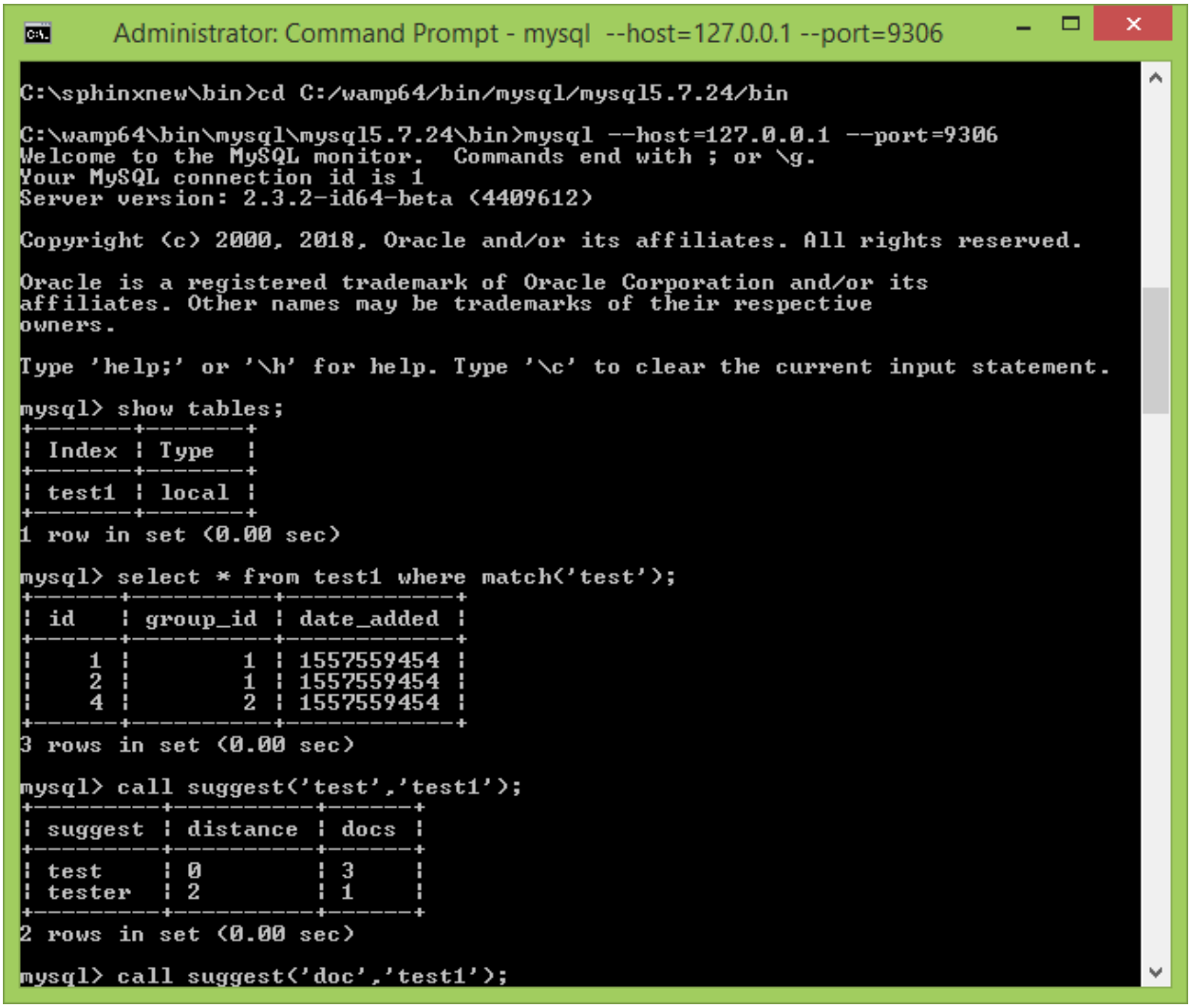
Figure 17:document table

Administrator: Command Prompt - mysql --host=127.0.0.1 --port=9306

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:/sphinx2/bin

C:\sphinx2\bin>indexer --rotate --all
Sphinx 2.0.7-id64-release (r3759)
Copyright (c) 2001-2012, Andrew Aksyonoff
Copyright (c) 2008-2012, Sphinx Technologies Inc (http://sphinxsearch.com)

using config file './sphinx.conf'...
indexing index 'test1'...
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
ERROR: index 'test1': rename C:/sphinx2/dat/test1.new.tmp.spd to C:/sphinx2/dat/
test1.new.new.spd failed: Broken pipe.
total 4 docs, 193 bytes
total 0.029 sec, 6584 bytes/sec, 136.46 docs/sec
indexing index 'idx1'...
WARNING: Attribute count is 0: switching to none docinfo
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 4 docs, 24 bytes
total 0.063 sec, 379 bytes/sec, 63.20 docs/sec
total 3 reads, 0.000 sec, 0.2 kb/call avg, 0.0 msec/call avg
total 15 writes, 0.001 sec, 0.1 kb/call avg, 0.0 msec/call avg
WARNING: could not open pipe (GetLastError()=2)
WARNING: indices NOT rotated.

C:\sphinx2\bin>search dan
Sphinx 2.0.7-id64-release (r3759)
Copyright (c) 2001-2012, Andrew Aksyonoff
Copyright (c) 2008-2012, Sphinx Technologies Inc (http://sphinxsearch.com)

using config file './sphinx.conf'...
index 'test1': query 'dan ': returned 0 matches of 0 total in 0.000 sec

words:
1. 'dan': 0 documents, 0 hits

index 'idx1': query 'dan ': returned 1 matches of 1 total in 0.000 sec

displaying matches:
1. document=1002, weight=1695
        id=1002
        name=dan
        mark=85

words:
1. 'dan': 1 documents, 1 hits
```

TASK 4:Autocomplete option and stemming words

Then we searched data on stemming words like singular/plural words and auto completion of words. The disadvantage of using sphinx search is that there's no support for 'did-you-mean', etc - although these can be done with other tools easily enough. Sphinx does stem words though using dictionaries, so 'walking' and 'walk' (for example) would be considered the same in searches.



```
Administrator: Command Prompt - mysql  --host=127.0.0.1 --port=9306

C:\sphinxnew\bin>cd C:/wamp64/bin/mysql/mysql5.7.24/bin

C:\wamp64\bin\mysql\mysql5.7.24\bin>mysql --host=127.0.0.1 --port=9306
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 2.3.2-id64-beta (4409612)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+-------+-------+
| Index | Type  |
+-------+-------+
| test1 | local |
+-------+-------+
1 row in set (0.00 sec)

mysql> select * from test1 where match('test');
+------+----------+------------+
| id   | group_id | date_added |
+------+----------+------------+
|    1 |        1 | 1557559454 |
|    2 |        1 | 1557559454 |
|    4 |        2 | 1557559454 |
+------+----------+------------+
3 rows in set (0.00 sec)

mysql> call suggest('test','test1');
+---------+----------+------+
| suggest | distance | docs |
+---------+----------+------+
| test    | 0        | 3    |
| tester  | 2        | 1    |
+---------+----------+------+
2 rows in set (0.00 sec)

mysql> call suggest('doc','test1');
```

Figure 18:auto complete option

Indexing a huge database

```
//source code
#
# Minimal Sphinx configuration sample (clean, simple, functional)
#

source src1
{
        type                    = mysql
```

```
        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = test
        sql_port                = 3306  # optional, default is 3306

        sql_query               = \
                SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
                FROM documents

        sql_attr_uint           = group_id
        sql_attr_timestamp      = date_added
}

source src1p0
{
   type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = table
        sql_port                = 3306  # optional, default is 3306

        sql_query               = \
                SELECT
OrderID,Region,Country,ItemType,SalesChannel,OrderPriority,OrderDate,ShipDate,UnitsSold,UnitPrice,
UnitCost,TotalRevenue,TotalCost,TotalProfit \
                FROM sample
                sql_range_step = 1000000

}

index test1
{
        source                  = src1
        path                    = C:/sphinxnew/dat/test1
        dict                    = keywords
        min_infix_len   = 10
}

index idxnew
{
        source                  = src1p0
        path                    = C:/sphinxnew/dat/idxnew.tmp
        dict                    = keywords
        min_infix_len   = 10
```

```
        docinfo         = extern
    charset_type        = utf-8

}

indexer
{
        mem_limit               = 128M
}


searchd
{
        listen                  = 9312
        listen                  = 9306:mysql41
        log                     = C:/sphinxnew/log/searchd.log
        query_log               = C:/sphinxnew/log/query.log
        read_timeout            = 5
        max_children            = 30
        pid_file        = C:/sphinxnew/log/searchd.pid
        seamless_rotate                 = 1
        preopen_indexes                 = 1
        unlink_old              = 1
        workers                         = threads # for RT to work
        binlog_path             = C:/sphinxnew/dat
}
```

Database to be indexed

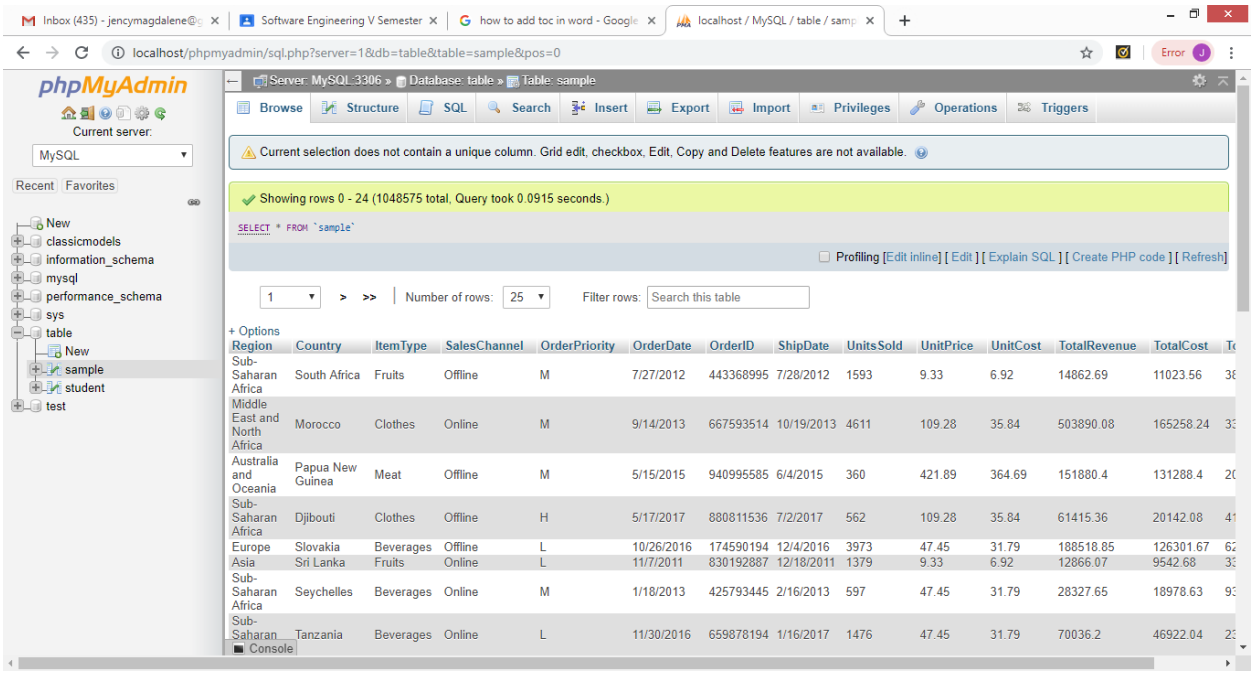


Figure 19:database with 1 million records

Output

```
Administrator: Command Prompt - mysql  --host=127.0.0.1 --port=9306
Bye

C:\wamp64\bin\mysql\mysql5.7.24\bin>cd C:/sphinxnew/bin

C:\sphinxnew\bin>indexer --rotate --all
Sphinx 2.3.2-id64-beta (4409612)
Copyright (c) 2001-2016, Andrew Aksyonoff
Copyright (c) 2008-2016, Sphinx Technologies Inc (http://sphinxsearch.com)

using config file './sphinx.conf'...
WARNING: key 'charset_type' was permanently removed from Sphinx configuration. R
efer to documentation for details.
indexing index 'test1'...
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 4 docs, 198 bytes
total 0.088 sec, 2242 bytes/sec, 45.29 docs/sec
indexing index 'idxnew'...
WARNING: Attribute count is 0: switching to none docinfo
collected 1048575 docs, 104.5 MB
sorted 25.3 Mhits, 100.0% done
total 1048575 docs, 104516376 bytes
total 23.445 sec, 4457876 bytes/sec, 44724.26 docs/sec
total 20 reads, 0.123 sec, 9909.5 kb/call avg, 6.1 msec/call avg
total 383 writes, 0.186 sec, 876.1 kb/call avg, 0.4 msec/call avg
rotating indices: successfully sent SIGHUP to searchd (pid=1852).

C:\sphinxnew\bin>cd C:/wamp64/bin/mysql/mysql5.7.24/bin

C:\wamp64\bin\mysql\mysql5.7.24\bin>mysql --host=127.0.0.1 --port=9306
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 2.3.2-id64-beta (4409612)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+--------+-------+
| Index  | Type  |
+--------+-------+
| idxnew | local |
| test1  | local |
+--------+-------+
2 rows in set (0.00 sec)
```

Figure 20:indexing

```
Administrator: Command Prompt - mysql  --host=127.0.0.1 --port=9306

mysql> select * from idxnew where match('fruits');
+-----------+
| id        |
+-----------+
| 100014054 |
| 100036799 |
| 100040233 |
| 100043666 |
| 100050532 |
| 100053966 |
| 100060832 |
| 100100743 |
| 100107610 |
| 100114476 |
| 100117909 |
| 100121343 |
| 100128209 |
| 100161254 |
| 100164687 |
| 100178420 |
| 100181853 |
| 100185286 |
| 100225198 |
| 100242364 |
+-----------+
20 rows in set (0.01 sec)

mysql> call suggest('fruit','idxnew');
+---------+----------+-------+
| suggest | distance | docs  |
+---------+----------+-------+
| fruits  | 1        | 79214 |
+---------+----------+-------+
1 row in set (0.00 sec)

mysql> _
```

Figure 21:searching

TASK 5

The main disadvantage in plain indexes is that it takes more time while indexing so to overcome this problem we use

1. RT indexes(real time)
2. Main and delta indexes

Real-Time Indexing (RT)

A Real-Time Index is split into two parts: one that always stay in memory, receiving new content; and a second that stays on disk, which is very similar to a plain index in structure. All new data goes to the RAM chunk. The size of this chunk is controlled by the rt_mem_limit configuration option. When this limit is reached, the RAM chunk is flushed to a disk chunk. A disk chunk is just like a plain index, the dictionary and stored attributes will be loaded in memory. After flushing, the RAM chunk is empty and can again be filled with data. The process repeats and a new disk chunk will be created. As we insert more data, more disk chunks will be created.This means the Sphinx daemon will need to hit more files on disk than in a normal plain index, which means more I/O. It then needs to merge results from all the chunks,

which translates, in the end, to lower search speeds. This kind of degradation is called, **'RT index fragmentation'**. In conclusion, the value of rt_mem_limitand the size of the data set will determine how many disk chunks are created. When the index becomes highly fragmented across many disk chunks, performance suffers. It's also important to remark that Sphinx will not use more memory than actually is necessary, so if the RT index only uses 1 MB while the limit is set to 2 GB, it will only consume 1 MB anyway.
Eliminating the I/O problem isn't everything because **searchd** still needs to go through several chunks and merge the results — CPU can be a bottleneck. So, realtime indexes lag behind the search speed of a plain index, which consists of a single piece. To bring realtime index performance close to plain index performance, it's necessary to **OPTIMIZE**. The optimization does nothing more than merge all the disk chunks into one. The operation is quite I/O intensive, as it needs to read all data from a disk chunk, create a temporary chunk (which isn't searchable) and merge the next chunk into it. After that, the temporary chunk is brought in and the chunks that have been merged are deleted.

Source code

```
index testrt
{

        type            = rt
        path            = C:/sphinx2/data/del
        rt_field        = name
        rt_field        = body
        rt_field        = tags
        rt_attr_string  = name
        rt_attr_uint    = post_type
}




searchd
{
        listen                  = 9312
        listen                  = 9306:mysql41
        log                     =  C:/sphinx2/log/searchd.log
        query_log               =  C:/sphinx2/log/query.log
        read_timeout            = 5
        max_children            = 30
        pid_file        =  C:/sphinx2/log/searchd.pid
        seamless_rotate                 = 1
        preopen_indexes                 = 1
        unlink_old              = 1
        workers                         = threads # for RT to work
        binlog_path             =  C:/sphinx2/data
```

}

Examples

```
Administrator: Command Prompt - mysql  --host=127.0.0.1 --port=9306
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+--------+-------+
| Index  | Type  |
+--------+-------+
| test1  | local |
| testrt | rt    |
+--------+-------+
2 rows in set (0.00 sec)

mysql> SELECT *, WEIGHT() FROM testrt WHERE MATCH('test');
+------+------+----------+
| id   | gid  | weight() |
+------+------+----------+
|    1 |  123 |     1356 |
|    2 |  234 |     1356 |
+------+------+----------+
2 rows in set (0.00 sec)

mysql> _
```

Figure 22: RT indexes

```
Administrator: Command Prompt - mysql  -h 127.0.0.1 -P9306
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+-------+------+
| Index | Type |
+-------+------+
| sop   | rt   |
+-------+------+
1 row in set (0.00 sec)

mysql> select * from sop;
Empty set (0.00 sec)

mysql> desc sop;
+-----------+--------+
| Field     | Type   |
+-----------+--------+
| id        | bigint |
| name      | field  |
| body      | field  |
| tags      | field  |
| post_type | uint   |
| name      | string |
+-----------+--------+
6 rows in set (0.00 sec)

mysql> insert into sop(1,'harry','student','happy',34);
ERROR 1064 (42000): sphinxql: syntax error, unexpected CONST_INT near '1,'harry'
,'student','happy',34)'
mysql> insert into sop(1,'harry','student','happy',34,'cool');
ERROR 1064 (42000): sphinxql: syntax error, unexpected CONST_INT near '1,'harry'
,'student','happy',34,'cool')'
mysql> insert into sop values(1,'harry','student','happy',34,'cool');
Query OK, 1 row affected (0.00 sec)

mysql> insert into sop values(1,'harry','student','happy',34);
ERROR 1064 (42000): column count does not match schema (expected 6, got 5)
mysql> insert into sop values(2,'dan','child','excited',62,'swag');
Query OK, 1 row affected (0.00 sec)

mysql> select * from sop;
+------+-----------+------+
| id   | post_type | name |
+------+-----------+------+
|    1 |        34 | cool |
|    2 |        62 | swag |
+------+-----------+------+
2 rows in set (0.00 sec)

mysql> _
```

Figure 23:rt index

TASK 6:Sorting the searches

Ranker option

```
Administrator: Command Prompt - mysql  --host=127.0.0.1 --port=9306
|     1 |  123 |      1356 |
|     2 |  234 |      1356 |
+-------+------+-----------+
2 rows in set (0.00 sec)

mysql> SELECT *, WEIGHT() FROM testrt WHERE MATCH('test')OPTION ranker=sph04;
+-------+------+-----------+
| id    | gid  | weight()  |
+-------+------+-----------+
|     1 |  123 |      6356 |
|     2 |  234 |      6356 |
+-------+------+-----------+
2 rows in set (0.00 sec)

mysql> SELECT *, WEIGHT() FROM testrt WHERE MATCH('test|one|two')OPTION ranker=s
ph04;
+-------+------+-----------+
| id    | gid  | weight()  |
+-------+------+-----------+
|     1 |  123 |     10500 |
|     2 |  234 |      6500 |
+-------+------+-----------+
2 rows in set (0.00 sec)

mysql> _
```

Figure 24:using ranker option

Difference between real and plain indexes

Plain Index

Bad:
- i have no mysql so i need to build an XMLpipe2 framework
- no real time.

Good
+ low memory consumption
+ proven technologie used for several years

RT index

Bad:
- memory consumption
- new technologie that just come out from the beta stage

Good

+ Real time index

+ sphinxQL or API to add/update/delete records

Main and delta indexing

```
#
# Minimal Sphinx configuration sample (clean, simple, functional)
#

source main
{
        type                    = mysql

        sql_host                = localhost
        sql_user                = root
        sql_pass                =
        sql_db                  = test
        sql_port                = 3306  # optional, default is 3306

        sql_query_pre = SET NAMES utf8
  sql_query_pre = REPLACE INTO sph_counter SELECT 1, MAX(id) FROM documents
  sql_query = SELECT id, title, content FROM documents \
    WHERE id<=( SELECT max_doc_id FROM sph_counter WHERE counter_id=1 )
}


source delta : main
{
  sql_query_pre = SET NAMES utf8
  sql_query = SELECT id, title, content FROM documents \
    WHERE id>( SELECT max_doc_id FROM sph_counter WHERE counter_id=1 )
                sql_attr_string=title

}

index test
{
        source                  = main
        path                    = C:/sphinxbeta/data/test
}

index delta : test
{
  source = delta
  path = C:/sphinxbeta/data/delta.tmp
}
```

```
indexer
{
        mem_limit               = 128M
}


searchd
{
        listen                  = 9312
        listen                  = 9306:mysql41
        log                     =  C:/sphinxbeta/log/searchd.log
        query_log               =  C:/sphinxbeta/log/query.log
        read_timeout            = 5
        max_children            = 30
        pid_file        =  C:/sphinxbeta/log/searchd.pid
        seamless_rotate                 = 1
        preopen_indexes                 = 1
        unlink_old              = 1
        workers                         = threads # for RT to work
        binlog_path             =  C:/sphinxbeta/data
}
```

CONCLUSION

Being a part of this internship I conclude that I learned about search engines and have a clear understanding about both lucene and sphinx .Both have their own advantages and disadvantages but I preferably like to sphinx because the indexing speed is fast and have more facilities like rt indexes to increase indexing speed.

Output

```
Administrator: Command Prompt - mysql  -h 127.0.0.1 -P9306

C:\sphinxbeta\bin>indexer --rotate --all
Sphinx 2.3.2-id64-beta (4409612)
Copyright (c) 2001-2016, Andrew Aksyonoff
Copyright (c) 2008-2016, Sphinx Technologies Inc (http://sphinxsearch.com)

using config file './sphinx.conf'...
indexing index 'test'...
WARNING: Attribute count is 0: switching to none docinfo
collected 7 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 7 docs, 228 bytes
total 0.037 sec, 6081 bytes/sec, 186.72 docs/sec
indexing index 'delta'...
WARNING: Attribute count is 0: switching to none docinfo
collected 0 docs, 0.0 MB
total 0 docs, 0 bytes
total 0.023 sec, 0 bytes/sec, 0.00 docs/sec
total 2 reads, 0.000 sec, 0.3 kb/call avg, 0.0 msec/call avg
total 16 writes, 0.001 sec, 0.1 kb/call avg, 0.0 msec/call avg
rotating indices: successfully sent SIGHUP to searchd (pid=5608).

C:\sphinxbeta\bin>cd C:/wamp64/bin/mysql/mysql5.7.24/bin

C:\wamp64\bin\mysql\mysql5.7.24\bin>mysql -h 127.0.0.1 -P9306
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 2.3.2-id64-beta (4409612)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from tst where match('funny');
ERROR 1064 (42000): unknown local index 'tst' in search request
mysql> select * from test where match('funny');
+------+
| id   |
+------+
|    8 |
+------+
1 row in set (0.00 sec)

mysql> exit;
Bye
```

Figure 25:main and delta index

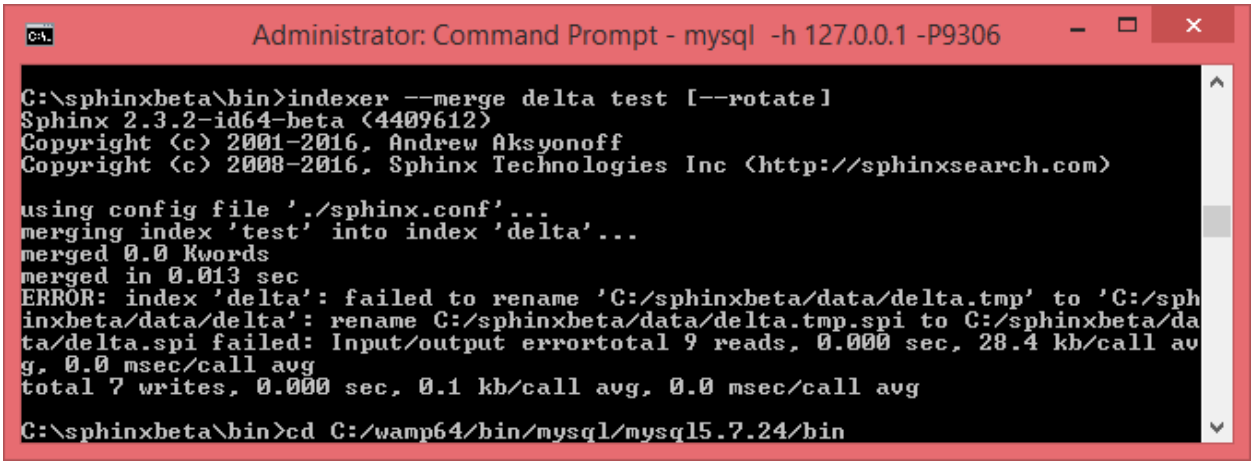


Figure 26;index merging

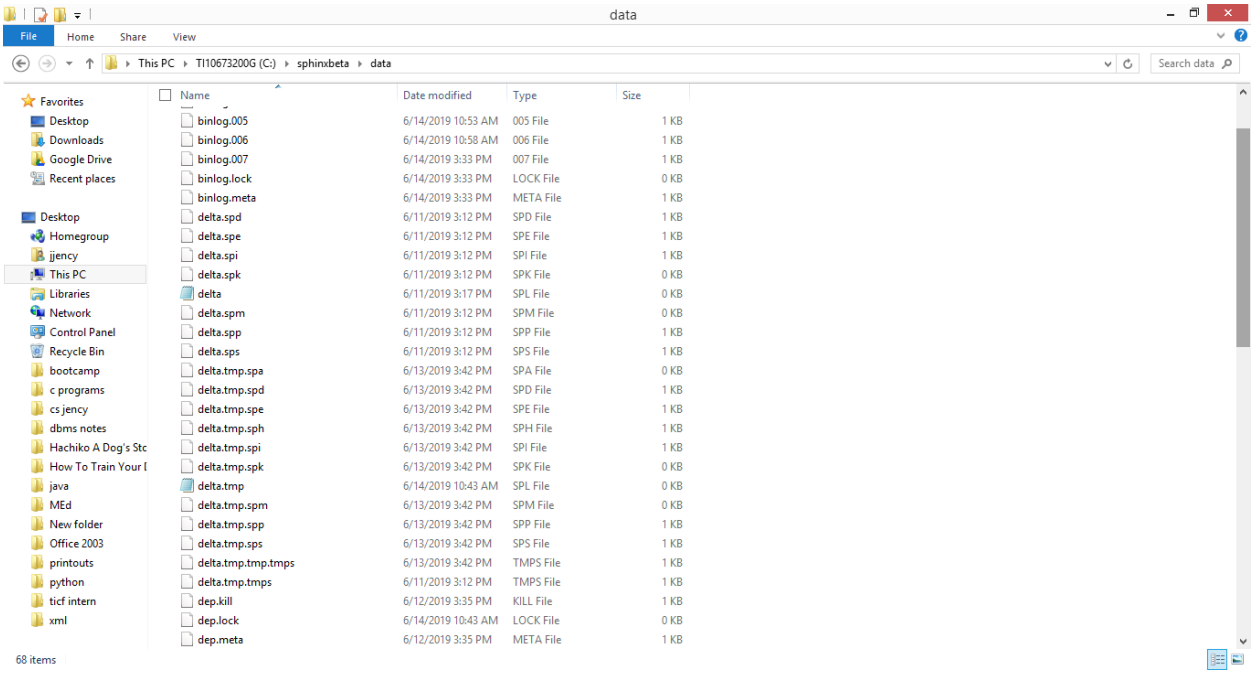


Figure 27:merged indexes