IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 10, October 2015.
www.ijiset.com
ISSN 2348 – 7968

Algorithm to change SQL to JAVA API for Elastic Search and SOLR

Sanjeeva Rao Nimmakayala[1], Dr. Srinivasa Rao T[2] and Dr.Srinivas G[3]

[1] Information Technology, Gitam University, Vishakhapatnam, Andhra Pradesh, India

[2] Information Technology, Gitam University, Vishakhapatnam, Andhra Pradesh, India

[3] Information Technology, Gitam University, Vishakhapatnam, Andhra Pradesh, India

Abstract

In this paper we proposed an Algorithm and API for changing SQL query to JAVA API for ElasticSearch[3] and SOLR[2]. From the 1990's people are more habituated with the sql queries and Now people are moving to Bigdata and trying to search based on Lucene queries. By this algorithm people can search directly based on SQL queries with JAVA API in Elastic search and SOLR[2] which are latest Bigdata search technologies developed based on LUCENE[1] Algorithm.

Keywords: *SOLR, ELASTIC Search, LUCENE.*

1. Introduction

Bigdata became more important since conventional database cannot provide the necessary efficiency and performance. SQL(Structured Query Language) is a programming language designed for managing data in a Relational database management system(RDMS). This is used from the 90's to search the data from the databases with giving sql queries. Day by day the members using internet getting increases with huge count, the Data base size also increasing very high. From this huge data retrieving the data became more challenges. To avoid the performance issue or to retrieve the data in less time Lucene algorithm is implemented.

Lucene Algorithm: It is an open-source Java full-text search library which makes it easy to add search functionality to an application or website.

SOLR: Apache Solr is a fast open-source Java search server. Solr enables you to easily create search engines which searches websites, databases and files. It was developed on the top of lucene algorithm.

ELASTICSearch: It is an Open Source (Apache 2), Distributed Search Engine built on top of Apache Lucene[1]. It allows you to start with one machine and scale to hundreds, and supports distributed search deployed over Amazon EC2's cloud hosting

2. Algorithm

1) Define the constants in the order of sql query from left to right like below
clauses= Select, From, where, Keyword, HINT, MATCH, SLOP, JOIN, FILTER, UNIQUE, DISTINCT, SORT
beginIndex = 0; val=null;

2) For loop i=CLAUSES.length - 1 and if i >= 0
beginIndex = dqlUpper.indexOf(CLAUSES[i])
if beginIndex >= 0
val = dql.substring(beginIndex + CLAUSES[i].length(), endIndex)

if (val != null && (val = val.trim()).length() > 0)
map.put(DQL_CLAUSES[i], val);
endIndex = beginIndex;
decrement I by 1;
end for loop

3. JAVA API OF ALGORITHM

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

ISSN 2348 – 7968

```
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.IOUtils;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;


class SQLToLUCENE{

 public static final String QUERY_SELECT = "SELECT ";
        public static final String QUERY_INSERT = "INSERT ";
        public static final String QUERY_UPDATE = "UPDATE ";
        public static final String QUERY_DELETE = "DELETE ";
        public static final String QUERY_FROM = " FROM ";
        public static final String QUERY_WHERE = " WHERE ";
        public static final String QUERY_KEYWORD = " KEYWORD ";
        public static final String QUERY_HINT = " HINT ";
        public static final String QUERY_MATCH = " MATCH ";
        public static final String QUERY_SLOP = " SLOP ";
        public static final String QUERY_JOIN = " JOIN ";
        public static final String QUERY_FILTER = " FILTER ";
        public static final String QUERY_UNIQUE = " UNIQUE ";
        public static final String QUERY_DISTINCT = " DISTINCT ";
        public static final String QUERY_SORT = " SORT ";
        public static final String QUERY_VAR = "VAR";
        public static final String QUERY_CONST = "CONST";
        public static final String QUERY_AS = "AS";
        public static final String QUERY_TO = "TO";

        public static final String[] CLAUSES = {
        QUERY_SELECT,
                QUERY_FROM,
                QUERY_WHERE,
                QUERY_KEYWORD,
                QUERY_HINT,
                QUERY_MATCH,
                QUERY_SLOP,
                QUERY_JOIN,
                QUERY_FILTER,
                QUERY_UNIQUE,
                QUERY_DISTINCT,
                QUERY_SORT };

        public static void main(String args[]) throws IOException{
String sqlQuery= “Provide the sql query here” ;
Map<String, String> map = new HashMap<String, String>();
int endIndex = sqlQuery.length();
int beginIndex = 0;
String val = null;
String sqlUpper = sqlQuery.toUpperCase();
for (int i = CLAUSES.length - 1; i >= 0; i--) {
beginIndex = sqlUpper.indexOf(CLAUSES[i]);
if (beginIndex >= 0) {
val = sqlQuery.substring(beginIndex + CLAUSES[i].length(), endIndex);
if (val != null && (val = val.trim()).length() > 0) {
map.put(CLAUSES[i], val);
}       endIndex = beginIndex;
                        }
                }
String indexName = map.get(QUERY_SELECT);

        }
```

3.2 Explanation of above API with example

Take example query as like below :

String dql="select a,b,c from tablename where ex=2 and test=3";

As per the algorithm of for loop it will check the condition of Clauses from the end. First it will get the ‘I’ value in the clause as SORT. It will check the clause available in sql query or not. If the condition fails it will get the next less value index from the clause.
As per the example query it will run the same condition up to ‘WHERE’,. Then it will do the substring after where to remaining of the query.

It will divide from substring as “ex=2 and test=3” .
It will put in the MAP as key = where and value =”ex=2 and test=3;

ISSN 2348 – 7968

It will update in the map in the same way for all the constants.
From elastic search and Solr API's now they can directly get the value in the programming just calling the map with key value like Map.get(`QUERY_SELECT`) or Map.get(`QUERY_WHERE`).

4. SUMMARY

We have demonstrated an algorithm to change the SQL query to the latest Bigdata searching technologies query for Elasticsearch and SOLR..

References

[1] Rujia Goa,Danying Li and Wanlong Li, " Application of Full Text Search Engine Based on Lucene ", Advances in Internet of Things, 2012,2,106-109
[2] http://lucene.apache.org/solr/.
[3] https://www.elastic.co/.