**BIG DATA PROGRAMMING**

**PROJECT\_BASED EXAM 🡪 SOLR USE CASE**

**TEAM MEMBERS AND CONTRIBUTION**

Roshini varada --Facebook Mutual Friends

Project link -- <https://github.com/RoshiniVarada/BDP_Projects/tree/master/Case1-FacebookMutualFriends>

Wiki link -- <https://github.com/RoshiniVarada/BDP_Projects/wiki/Case1-FacebookMutualFriends>

Zakari, Abdulmuhaymin --Youtube analysis

Project link -- <https://github.com/RoshiniVarada/BDP_Projects/tree/master/Case2-YouTubeAnalysis>

Wiki link -- <https://github.com/RoshiniVarada/BDP_Projects/wiki/Case2-YouTubeAnalysis>

Sarika Reddy Kota -- Hive Use case

Project link -- <https://github.com/RoshiniVarada/BDP_Projects/tree/master/Case3_HiveUseCase>

Wiki link -- <https://github.com/RoshiniVarada/BDP_Projects/wiki/Case3_Hive_Usecase>

Pallavi Arikatla -- Solr use case

Project link -- <https://github.com/RoshiniVarada/BDP_Projects/wiki/Case4-Solr>

Wiki link -- <https://github.com/RoshiniVarada/BDP_Projects/wiki/Case4-Solr>

**OBJECTIVE:**

Implementation of queries in Apache Solr using Apache lucene.

**INTRODUCTION:**

* Solr is a search platform that is built using Lucene which is truly compatible and highly scalable.
* Solr indexes and stores the documents.
* As soon as we search for a particular keyword, it hits the solr and returns every possible relatable result searching from several other websites.

**QUESTION 4: (1)**

**IDEA OF THE PROJECT:**

Idea of the project is to apply concepts learnt in Big-Data-Programming so far in real-time. Hadoop is a open source platform to store immense amount of data and to run numerous applications on cluster commodities. In this project we apply different concepts like MapReduce, Hive and Solr to work on that massive data and make task simpler.

**QUESTION 4: (2)**

**USAGE OF PROJECT IN REAL-TIME:**

Apache Solr is one such open source platform which provides search and index replication. It is built using Apache lucene which makes it scalable. It works for multiple sites, because it can index the entire data and search the information in several websites and returns all the relevant recommendations for the searched content. Solr aids you to effortlessly create search engines which searches websites, databases and documents. Hence it is an effective way to work on large data in real-time.

**QUESTION 4: (3)**

**IMPLEMENTATION:**

* Open a new terminal in cloudera.
* Create schema configuration for the selected dataset.

solrctl instancedir --generate /tmp/hero4

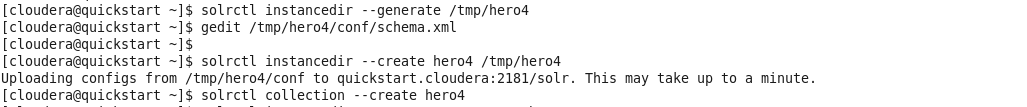
gedit /tmp/hero4/conf/schema.xml

* A schema page pops up which is to be edited. Replace all the id's with the id's present in the dataset and change the unique id with first id in the dataset.
* Now we have to create this dataset in the Solr interface. Type the following commands which helps in pushing the dataset to Solr page.

Create instance directory:

solrctl instancedir --create hero4 /tmp/hero4

solrctl collection --create hero4



\* Similarly now apply this to another dataset if required. And the commands are as follows:

Create schema configuration:

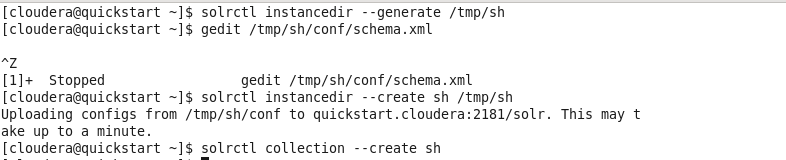
solrctl instancedir --generate /tmp/sh

/tmp/sh/conf/schema.xml

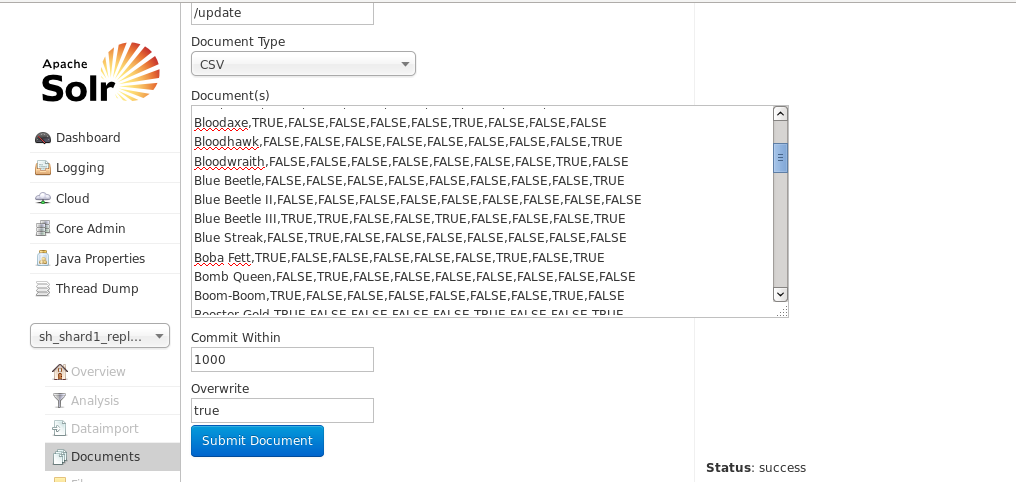
Create instance directory:

solrctl instancedir --create sh /tmp/sh

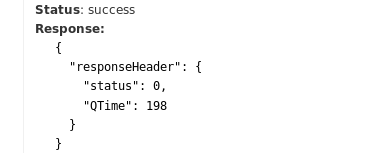
solrctl collection --create sh



* Now get to Solr interface and select the dataset created.
* Click on documents --> Change file type (CSV) --> Copy the entire data in the dataset and place it under documents column --> Sumbit the document.



* If your document import is successful you can observe the following after submitting the document.



* Now you are ready to run the queries.

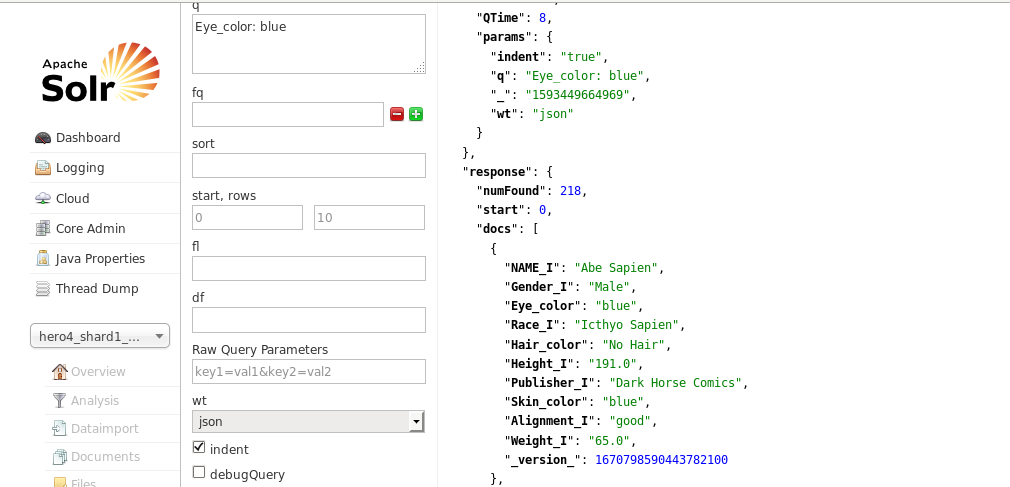
**QUERIES:**

**Query 1:**

Keyword matching:

Searches for a particular word in the entire index and displays that data present in the entire dataset.

Eye\_color: blue

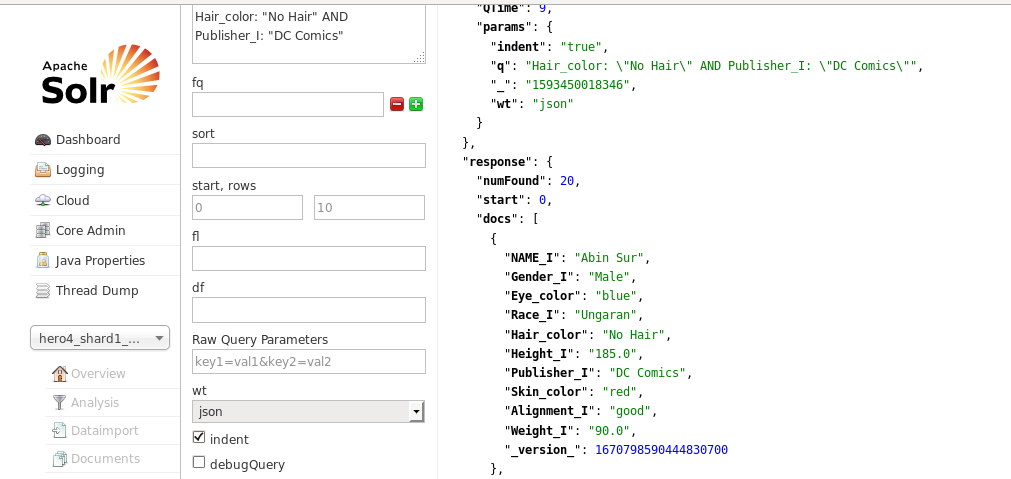


**Query 2:**

AND: Searches for both the phrases.

Hair\_color: "No Hair" AND Publisher\_I: "DC Comics"

Searches for 'No Hair' in Hair\_color index and 'DC Comics' in Publisher\_I index simultaneously.

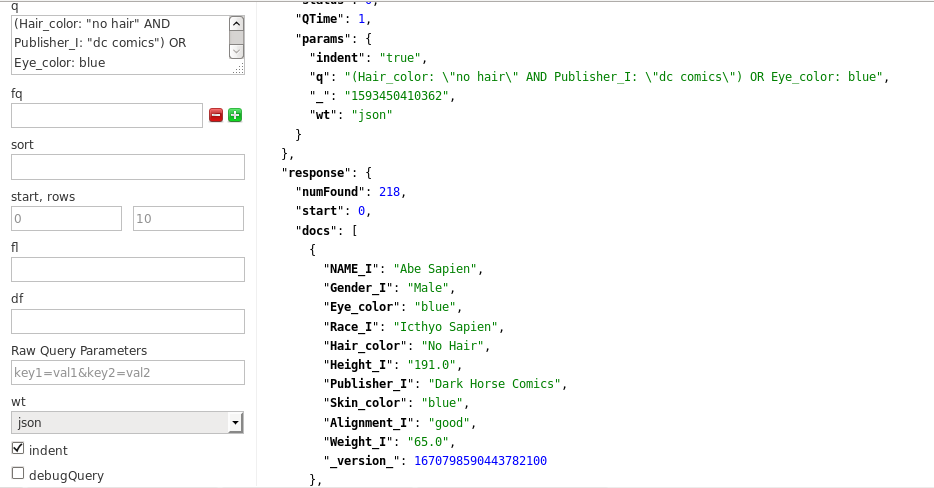


**Query 3:**

OR: Searches for either of the phrases.

Hair\_color:"No Hair" AND Publisher\_I:"DC Comics" OR Eye\_color:blue

Searches for 'No Hair' in Hair\_color index and 'DC Comics' in Publisher\_I index OR 'Blue' in Eye\_color index.

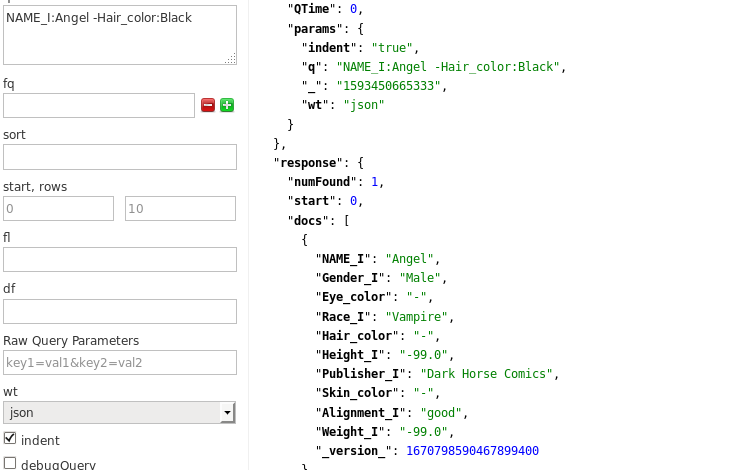


**Query 4:**

NOT: Searches for a keyword in a index with a NOT limitation.

NAME\_I:Angel -Hair\_color:black

Searches for 'Angel' in index NAME\_I without 'black' as Hair\_color in that row.



**Query 5:**

Wildcard: Searches for a particular word start in a string.

Eye\_color:yellow\*

Searches for any word starting with 'yellow' in index Eye\_color.

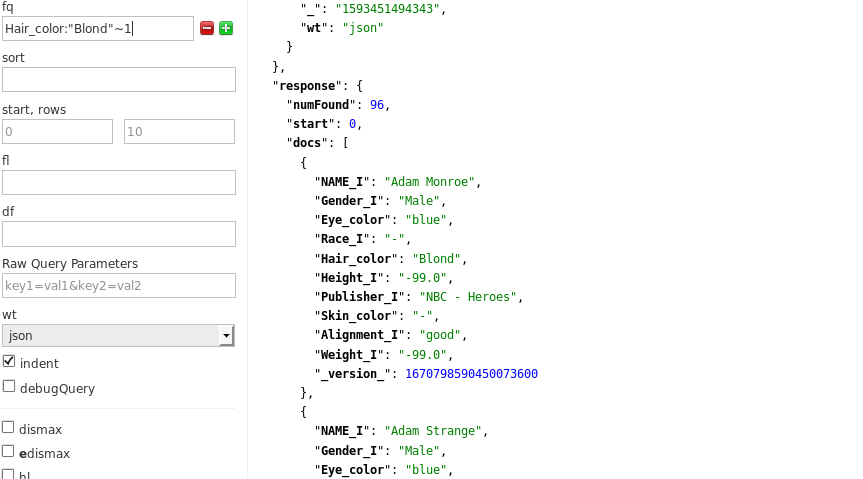


**Query 6:**

Proximity: Searches for word within proximity range.

Hair\_color:"Blond"~1

Searches for 'Blond' in Hair\_color within 1 word from each other.



**Query 7:**

Boolean multiple clauses: A Boolean query contains multiple queries to work on.

(+agility\_I:TRUE +agility\_I:(FALSE faceting)) OR (+Lantern\_Power\_Ring:TRUE +Lantern\_Power\_Ring:(TRUE faceting))

OR operation means that the clauses are optional.



**Query 8:**

Boosted: Creates importance of for a particular searched word. It prioritizes the searched content as per the boosting factor.

Dimensional\_Awareness:(TRUE^10 TRUE)

It prioritizes TRUE to ^10 times in Dimensional\_Awareness. The operand '^' helps in prioritizing the content.



**Query 9**:

Range: Allows to match fields and displays output with in the specifies upper and lower limit.

Weight\_I: [95.0 TO \*]

The above query helps in printing the data containing Weight\_I above 95.0.



**Query 10:**

Facet: Determines unique terms in a specified field.

&facet=True

&facet.field=Skin\_color

Displays only Skin\_color field.

&facet=Female

&facet.field=Height\_I

Displays only Height\_I field.





Time of execution:

As I have worked on single dataset at a time, execution takes no mean time. And the execution time usually be dependent on the indexes result.

Query 1: 1.83ms

Query 2: 2.45ms

Query 3: 3.02ms

Query 4: 2.55ms

Query 5: 2.85ms

Query 6: 3.38ms

Query 7: 4.61ms

Query 8: 4.26ms

Query 9: 4.91ms

Query 10: 5.61ms

Execution of all these queries can happen in less than a minute. Hence this can be an effective word search method.

**QUESTION 4: (4)**

**CHALLENGES FACED:**

I have almost faced no challenges during the process of execution. Entire procedure went efficiently without any obstacles. The only issue I have faced is while loading Datasets to the Solr interface. Although the issue is not so problematic, it consumes time if your schema isn’t edited properly. Paying attention while configuring schema for large dataset helps.

**QUESTION 4: (5)**

**MILESTONES AND INTEGRATION OF THE PROJECT**:

As there is no dependency with each other’s split up work, the work went effective without any issues and we have successfully accomplished the tasks taken.