**Inverted Index Project**

Pattern Description

The inverted index pattern is commonly used as an example for MapReduce analytics. We’re going to discuss the general case where we want to build a map of some term to a list of identifiers.

Intent

Generate an index from a data set to allow for faster searches or data enrichment capabilities.

Motivation

It is often convenient to index large data sets on keywords, so that searches can trace terms back to records that contain specific values. While building an inverted index does require extra processing up front, taking the time to do so can greatly reduce the amount of time it takes to find something.

Search engines build indexes to improve search performance. Imagine entering a keyword and letting the engine crawl the Internet and build a list of pages to return to you. Such a query would take an extremely long amount of time to complete. By building an inverted index, the search engine knows all the web pages related to a keyword ahead of time and these results are simply displayed to the user. These indexes are often ingested into a database for fast query responses. Building an inverted index is a fairly straightforward application of MapReduce because the framework handles a majority of the work.

Applicability

Inverted indexes should be used when quick search query responses are required. The results of such a query can be preprocessed and ingested into a database.

Structure

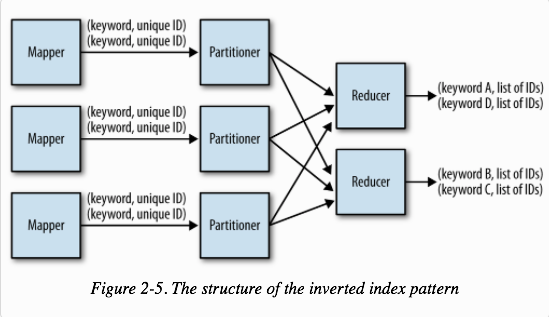
Figure 2-5 shows the general structure of how an inverted index is executed in MapReduce. The breakdown of each MapReduce component is described in detail below:

The mapper outputs the desired fields for the index as the key and the unique identifier as the value.

The combiner can be omitted if you are just using the identity reducer, because under those circumstances a combiner would just create unnecessary processing. Some implementations concatenate the values associated with a group before outputting them to the file system. In this case, a combiner can be used. It won’t have as beneficial an impact on byte count as the combiners in other patterns, but there will be an improvement.

The partitioner is responsible for determining where values with the same key will eventually be copied by a reducer for final output. It can be customized for more efficient load balancing if the intermediate keys are not evenly distributed.

The reducer will receive a set of unique record identifiers to map back to the input key. The identifiers can either be concatenated by some unique delimiter, leading to the output of one key/value pair per group, or each input value can be written with the input key, known as the identity reducer.



Consequences

The final output of is a set of part files that contain a mapping of field value to a set of unique IDs of records containing the associated field value.

Performance analysis

The performance of building an inverted index depends mostly on the computational cost of parsing the content in the mapper, the cardinality of the index keys, and the number of content identifiers per key.

Parsing text or other types of content in the mapper can sometimes be the most computationally intense operation in a MapReduce job. This is especially true for semi-structured data, such as XML or JSON, since these typically require parsing arbitrary quantities of information into usable objects. It’s important to parse the incoming records as efficiently as possible to improve your overall job performance.

If the number of unique keys and the number of identifiers is large, more data will be sent to the reducers. If more data is going to the reducers, you should increase the number of reducers to increase parallelism during the reduce phase.

Inverted indexes are particularly susceptible to hot spots in the index keys, since the index keys are rarely evenly distributed. For example, the reducer that handles the word “the” in a text search application is going to be particularly busy since “the” is seen in so much text. This can slow down your entire job since a few reducers will take much longer than the others. To avoid this problem, you might need to implement a custom partitioner, or omit common index keys that add no value to your end goal.

Inverted Index Example

Wikipedia reference inverted index

Building an inverted index is a straightforward MapReduce application and is often the second example newcomers to MapReduce experience after the word count application. Much like the word count application, the bulk of the operation is a group and is therefore handled entirely by the MapReduce framework.

Suppose we want to add StackOverflow links to each Wikipedia page that is referenced in a StackOverflow comment. The following example analyzes each comment in StackOverflow to find hyperlinks to Wikipedia. If there is one, the link is output with the comment ID to generate the inverted index. When it comes to the reduce phase, all the comment IDs that reference the same hyperlink will be grouped together. These groups are then concatenated together into a white space delimited String and directly output to the file system. From here, this data file can be used to update the Wikipedia page with all the comments that reference it.

The following descriptions of each code section explain the solution to the problem.

Problem: Given a set of user’s comments, build an inverted index of Wikipedia URLs to a set of answer post IDs .

Mapper code

The mapper parses the posts from StackOverflow to output the row IDs of all answer posts that contain a particular Wikipedia URL. First, the XML attributes for the text, post type, and row ID are extracted. If the post type is not an answer, identified by a post type of “2”, we parse the text to find a Wikipedia URL. This is done using the getWikipediaURL method, which takes in a String of unescaped HTML and returns a Wikipedia URL if found, or null otherwise. The method is omitted for brevity. If a URL is found, the URL is output as the key and the row ID is output as the value.

public static class WikipediaExtractor extends Mapper<Object, Text, Text, Text> {

private Text link = new Text();

private Text outkey = new Text();

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

Map<String, String> parsed = MRDPUtils.transformXmlToMap(value

.toString());

// Grab the necessary XML attributes

String txt = parsed.get("Body");

String posttype = parsed.get("PostTypeId");

String row\_id = parsed.get("Id");

// if the body is null, or the post is a question (1), skip

if (txt == null || (posttype != null && posttype.equals("1"))) {

return;

}

// Unescape the HTML because the SO data is escaped.

txt = StringEscapeUtils.unescapeHtml(txt.toLowerCase());

link.set(getWikipediaURL(txt));

outkey.set(row\_id);

context.write(link, outkey);

}

}

Reducer code

The reducer iterates through the set of input values and appends each row ID to a String, delimited by a space character. The input key is output along with this concatenation.

public static class Concatenator extends Reducer<Text,Text,Text,Text> {

private Text result = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

boolean first = true;

for (Text id : values) {

if (first) {

first = false;

} else {

sb.append(" ");

}

sb.append(id.toString());

}

result.set(sb.toString());

context.write(key, result);

}

}

Combiner optimization

The combiner can be used to do some concatenation prior to the reduce phase. Because all row IDs are simply concatenated together, the number of bytes that need to be copied by the reducer is more than in a numerical summarization pattern. The same code for the reducer class is used as the combiner.”

Excerpt From: Donald Miner and Adam Shook. “MapReduce Design Patterns.”

Now…with all of this project background we can discuss the actual project below!

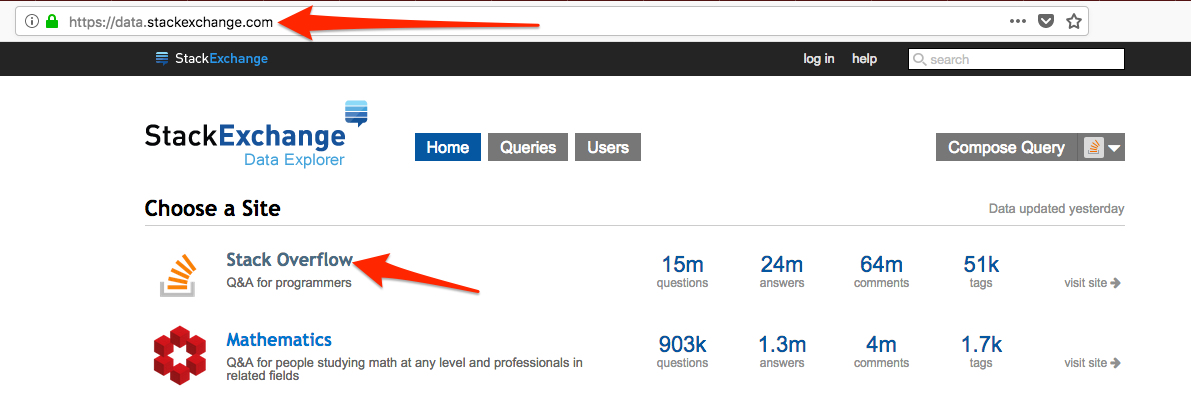
# MSDS 610 Spark Project

We are going to create an inverted index similar to what is shown above. We will be using Stack Overflow data but not Wikipedia.

## Problem:

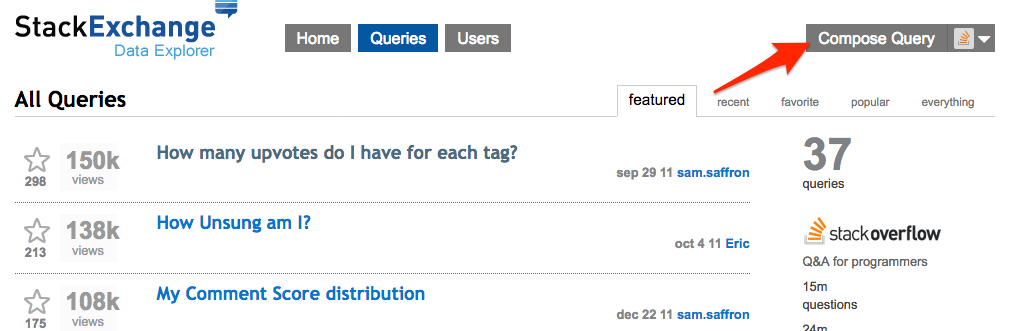
Query Stack Overflow’s data API to create and download a dataset. Next, use Spark, Jupyter notebook, and your cluster to create an inverted index listing all the Post IDs where each tag appears. To limit the query time and data transfer, please only use data from one year (e.g. 2017).

Stack Overflow publicly shares their data at https://data.stackexchange.com/.

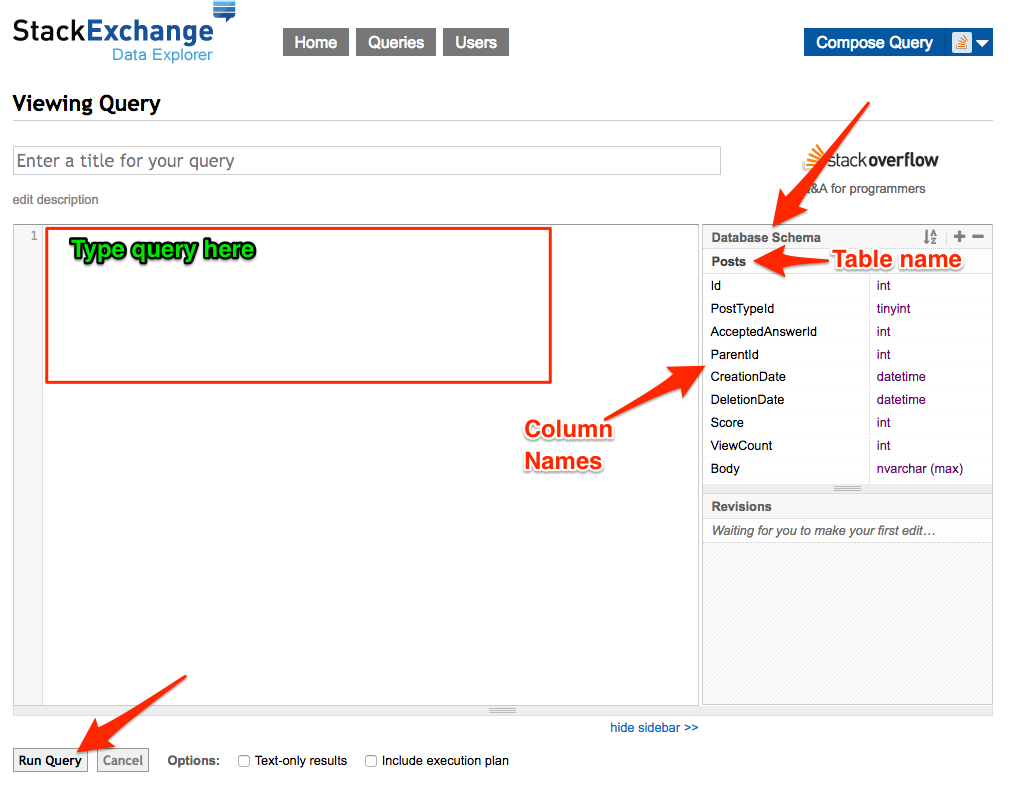


Following the Stack Overflow link will present an interface where you can see other peoples’ saved queries. Feel free to investigate them to get ideas.

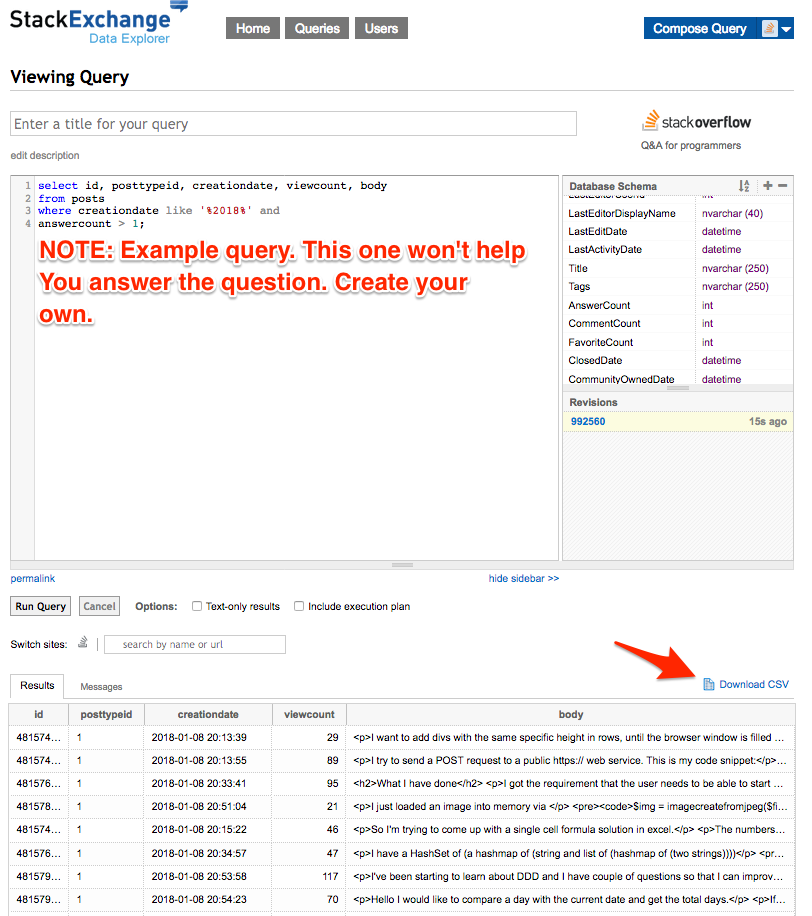
Eventually, you will want to “Compose Query.” Clicking the link will take you to a page where you can see table schemas, enter queries and see results:



You are encouraged to explore the tables and data using this interface. What tables and columns do you think will help you build the inverted index?



When you have the columns you want, click the “Download CSV” link to save the data as a file.



What do you do with this file?

At this point you are ready to load the data into Spark and start Mapping and Reducing to get the inverted index. All of your work will be performed in Junyper Notebooks. Remember to also review the Spark-Wordcount.ipynb file (download it from WorldClass and open it with Junyper Notebooks) for an example!

Bonus: Perhaps there are other questions you could answer:

* Which 5 tags had the most closed issues?
* What were the 5 most popular tags?
* Etc.

## Deliverable

Submit the output file that represents the inverted index as well as a short report that discusses the question, process for answering the question, and reflections or lessons learned in the process.