**Web Page Management Using Indexes**

**Introduction:**

# Indexing in Databases

Majestic Million CSV (https://blog.majestic.com/development/majestic-million-csv-daily/) is an example of a million record databases. According to the website, it “a list of the top 1 million website in the world, ordered by the number of referring subnets.”

As you might imagine, putting this data in a database table, and performing some database queries on it would be interesting. But as you also know, we might get a performance hit when using such data. The solution would be to indexing some of the field.

# Web Programing with NodeJS

We use Nodejs, in addition to several modules, in order to query database tables, and return data as a webpage. More in following sections

# Measuring Execution Time.

1. In sqlplus ORACLE utility, you can measure the execution of a single query by executing “set timing in”. See Bfor syntax help.
2. In \*nix (Unix/Linux), you can measure execution time of single commands using the command “time”, followed by the command to execute. See C for how to use it. In this project, we will use “real time” as measurement for the execution time.

## Implementation steps:

## Part 1: Indexing Business

1. Download Majestic Million CSV file, and save it in your server. You will call it majestic.csv
2. Using “more” utility, take a look into the file, and build an appropriate sql schema for it. Write the schema creation file in a file called “majestic\_schema.sql”. The schema file should create four identical tables called “MAJESTIC”, “MAJESTIC\_INDEX1”, “MAJESTIC\_INDEX2”, and “MAJESTIC\_INDEX3”.
3. Add to “majestic\_schema.sql” sql statements to create an index on the “TLD” field of table “MAJESTIC\_INDEX1”, field “RefSubNet” of table “MAJESTIC\_INDEX2”, and both “TLD” and “RefIPs” fields of table “MAJESTIC\_INDEX3”,
4. Load the data from the CSV file downloaded from Majestic Million CSV to all 4 tables created in 3). Use “time” Unix command to measure execution time. Put the command with time in a script called load\_data.
5. Write the following sql queries (each one in a .sql file), and run them on each table, while recording the execution time. Comment on the execution time differences.
   1. Return domain name, global rank and RedSubNet for all record where the domain is based in India (.in): query\_india\_domains.sql
   2. Return all sites (domain names) that they have 100k or more RefSubNet: query\_top\_domains.sql
   3. A query that returns the ranking of “wikipedia.org”:

query\_wikipedia.sql

1. Write a sql script to delete the tables, index files and data loaded part of this exercise. This is very important. Make sure that you execute it when done with this assignment. clean\_data.sql

Part 2: Lets put it on the Web:

We are going to program a web interface for our Majestic Million database, using NodeJS, HTML, CSS

1. Program the artifacts (both entry and view pages) to let the user query a domain name, then get its ranking. If domain does not exist, need to return an appropriate message
2. Program the artifacts (both entry and view pages) to let the user query domains belong with certain TLD. For example, if the user input India (.in), the page will return the GlobalRank and Domain name of each domain matching that TLD.
3. For each for the queries above, make a separate context for hitting each of the 4 tables created.

Part 3: Testing how fast if your website?

Now that the middle tier is built, lets try to test it.

- **CURL**

“curl” D is a \*nix utility that allows sending HTTP POST/GET to websites. For example, if you execute the following command, testing “bookapp.js”:

curl --data "author=J. K. Rowling" <url>

you will get the following reply:

<h4>Book Info:</h4>

<ul>

<li>Book Title: Harry Potter and the goblet of fire</li>

<li>Book Author: J. K. Rowling</li>

<li>Book Price: 25.95</li>

</ul>

<form action="/author" method="get">

<br><br>

<input type="submit" value="Search again">

</form>

1- Execute “curl” with “time” to measure POST requests for each query developed in part2. Timing data is collected and commented on it in the project README: query1\_curl, query2\_curl, query3\_curl

## - Apache JMeter

Measuring one request is something, but making the website can handle several concurrent clients is another. Imagine the load that Amazon.com handles everyday. Can your website handle that load?

Apache JMeter is a tool that allows load testing web sites. It is graphical tool that can, among other things:

* Automate the entry of HTTP POST parameters, then issue an HTTP post to a web page
* Simulate multiple users accessing the same web page.
* Collect results and response times.

We want to use JMeter to test our newly built web interface built in Part2. Some facts about the desired use of JMeter:

1. Install JMeter in your laptop/lab machine and test load with different number of users.

References:

A <http://www.java2s.com/Tutorial/Oracle/0180> Index/CreatinganIndex.htm

B <http://www.dba-oracle.com/t_measure_sql_response_time.htm>

C https://en.wikipedia.org/wiki/Time\_(Unix)

D https://en.wikipedia.org/wiki/CURL

E <http://jmeter.apache.org/usermanual/build-web-test-plan.html>