**PART-1**

Execute majestic\_schema.sql to create the below tables:

“MAJESTIC”, “MAJESTIC\_INDEX1”, “MAJESTIC\_INDEX2”, and “MAJESTIC\_INDEX3”

Load data from the majestic.csv file using the below control files:

majestic.ctl

majestic\_index1.ctl

majestic\_index2.ctl

majestic\_index3.ctl

**Time taken to load data in each table:**

Time taken to load the data in table MAJESTIC:

Elapsed time was: 00:03:00.73

CPU time was: 00:00:16.96

Time taken to load the data in table MAJESTIC\_INDEX1:

Elapsed time was: 00:03:08.98

CPU time was: 00:00:16.44

Time taken to load the data in table MAJESTIC\_INDEX2:

Elapsed time was: 00:03:19.78

CPU time was: 00:00:16.82

Time taken to load the data in table MAJESTIC\_INDEX3:

Elapsed time was: 00:03:16.76

CPU time was: 00:00:16.56

**Comment:** Time taken to load data in the table with index is more than the time taken to load data without index. Updating a table with indexes usually takes more time than updating a table without because the indexes also need an update.

**SQL queries : created 3 queries for 4 tables with names as below:**

**For table MAJESTIC:**

query\_india\_domains.sql

query\_top\_domains.sql

query\_wikipedia.sql

**For table MAJESTIC\_INDEX1:**

query\_india\_domains1.sql

query\_top\_domains1.sql

query\_wikipedia1.sql

**For table MAJESTIC\_INDEX2:**

query\_india\_domains2.sql

query\_top\_domains2.sql

query\_wikipedia2.sql

**For table MAJESTIC\_INDEX3:**

query\_india\_domains3.sql

query\_top\_domains3.sql

query\_wikipedia3.sql

**Execution time for query 1:**

1. Return domain name, global rank and RedSubNet for all record where the domain is based in India (in): query\_india\_domains.sql

Table MAJESTIC:

Elapsed time was: 00:00:02.11

Table MAJESTIC\_INDEX1:

Elapsed time was: 00:00:00.18

Table MAJESTIC\_INDEX2:

Elapsed time was: 00:00:01.88

Table MAJESTIC\_INDEX3:

Elapsed time was: 00:00:00.58

**Execution time for query 2:**

1. Return all sites (domain names) that they have 100k or more RefSubNet:

query\_top\_domains.sql

Table MAJESTIC:

Elapsed time was: 00:00:00.06

Table MAJESTIC\_INDEX1:

Elapsed time was: 00:00:00.06

Table MAJESTIC\_INDEX2:

Elapsed time was: 00:00:00.01

Table MAJESTIC\_INDEX3:

Elapsed time was: 00:00:00.07

**Execution time for query 3:**

1. A query that returns the ranking of “wikipedia.org”: query\_wikipedia.sql

Table MAJESTIC:

Elapsed time was: 00:00:00.00

Table MAJESTIC\_INDEX1:

Elapsed time was: 00:00:00.00

Table MAJESTIC\_INDEX2:

Elapsed time was: 00:00:00.01

Table MAJESTIC\_INDEX3:

Elapsed time was: 00:00:00.01

**Comment:**

Table MAJESTIC has no index, “MAJESTIC\_INDEX1” has index on “TLD” field, “MAJESTIC\_INDEX2” has index on field “RefSubNet”, and “MAJESTIC\_INDEX3” has index on both “TLD” and “RefIPs” fields.

The whole point of having an index is to speed up search queries by essentially cutting down the number of records/rows in a table that need to be examined.

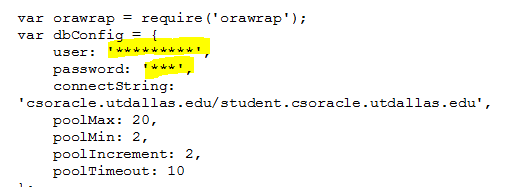
**In Query 1** we are searching for domain name, global rank and RedSubNet for all record where the domain is based in India (in). Since MAJESTIC\_INDEX1 table has index on “TLD”, so searching using table index “TLD” is taking the least time as the entire table does not have to be searched.

**In Query 2** we are searching for all sites that have 100k or more RefSubNet. Since MAJESTIC\_INDEX2 table has index on “RefSubNet”, so searching using table index is taking the least time as the entire table does not have to be searched.

**In Query 3** we are not searching using table index, so there is not much difference in the time.

**PART-2**

**Please execute majestic.js file . You need to replace the usename and password in the file to your credentials.**



Query 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

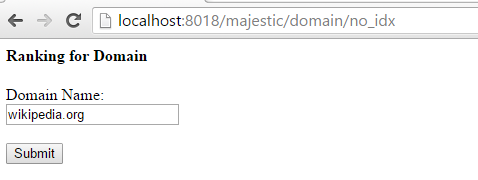
Note: Xming used to access web pages! (Use csgrads1.utdallas.edu instead of localhost incase using from other browsers)

**URLs for query 1:**

Query hitting table MAJESTIC with no index.

<http://localhost:8018/majestic/domain/no_idx>

Example to execute:



Query hitting table MAJESTIC\_INDEX1 with index on field “TLD”.

http://localhost:8018/majestic/domain/idx1

Query hitting table MAJESTIC\_INDEX2 with index on field “RefSubNet”

http://localhost:8018/majestic/domain/idx2

Query hitting table MAJESTIC\_INDEX3 with index on field both “TLD” and “RefIPs”.

http://localhost:8018/majestic/domain/idx3

QUERY2: 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.

**URLs for query 2:**

Query hitting table MAJESTIC with no index.

http://localhost:8018/majestic/domain2/no\_idx

Example:



Query hitting table MAJESTIC\_INDEX1 with index on field “TLD”.

http://localhost:8018/majestic/domain2/idx1

Query hitting table MAJESTIC\_INDEX2 with index on field “RefSubNet”

http://localhost:8018/majestic/domain2/idx2

Query hitting table MAJESTIC\_INDEX3 with index on field both “TLD” and “RefIPs”.

<http://localhost:8018/majestic/domain2/idx3>

**NOTE: Please change the port number in majestic.js file and all the URL’s if the port is busy!!**

**PART-3**

**Execute the following time curl commands for Query 1 in part 2**

**Query hitting table MAJESTIC with no index.**

time curl --data "domain\_noidx=wikipedia.org" <http://localhost:8018/search_noidx>

**Time taken:**

real 0m0.235s

user 0m0.002s

sys 0m0.004s

**Query hitting table MAJESTIC\_INDEX1 with index on field “TLD”.**

time curl --data "domain\_idx1=wikipedia.org" <http://localhost:8018/search_idx1>

**Time taken:**

real 0m0.232s

user 0m0.002s

sys 0m0.004s

**Query hitting table MAJESTIC\_INDEX2 with index on field “RefSubNet”**

time curl --data "domain\_idx2=wikipedia.org" <http://localhost:8018/search_idx2>

**Time taken:**

real 0m0.233s

user 0m0.003s

sys 0m0.002s

**Query hitting table MAJESTIC\_INDEX3 with index on field both “TLD” and “RefIPs”.**

time curl --data "domain\_idx3=wikipedia.org" <http://localhost:8018/search_idx3>

**Time taken:**

real 0m0.238s

user 0m0.003s

sys 0m0.002s

**Execute the following time curl commands for Query 2 in part 2**

**Query hitting table MAJESTIC with no index.**

time curl --data "domain2\_noidx=.in" <http://localhost:8018/search2_noidx>

**Time taken:**

real 0m0.282s

user 0m0.003s

sys 0m0.003s

**Query hitting table MAJESTIC\_INDEX1 with index on field “TLD”.**

time curl --data "domain2\_idx1=.in" <http://localhost:8018/search2_idx1>

**Time taken:**

real 0m0.249s

user 0m0.000s

sys 0m0.006s

**Query hitting table MAJESTIC\_INDEX2 with index on field “RefSubNet”**

time curl --data "domain2\_idx2=.in" <http://localhost:8018/search2_idx2>

**Time taken:**

real 0m0.286s

user 0m0.003s

sys 0m0.003s

**Query hitting table MAJESTIC\_INDEX3 with index on field both “TLD” and “RefIPs”.**

time curl --data "domain2\_idx3=.in" <http://localhost:8018/search2_idx3>

**Time taken:**

real 0m0.280s

user 0m0.004s

sys 0m0.002s

**Comment:**

(HTTP) Sends the specified data in a POST request to the HTTP server, in the same way that a browser does when a user has filled in an HTML form and presses the submit button. This will cause curl to pass the data to the server.

**In Query 1** user query a domain name, then get its ranking. Here domain name is not an index for any of the tables, so there is no much difference in the time taken for the query to fetch data from the database.

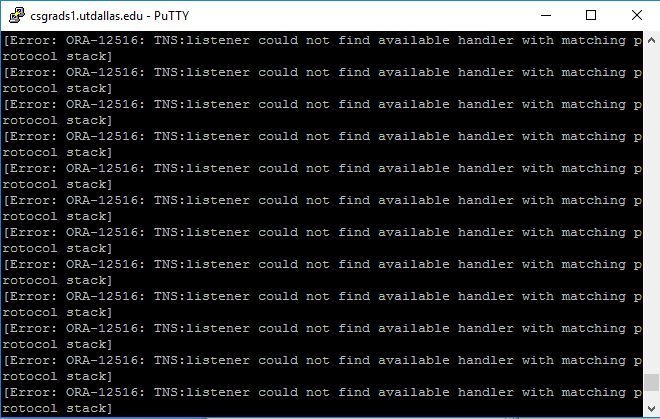
**In Query 2** user query domains belong with certain TLD (e.g. in) the page will return the GlobalRank and Domain name of each domain matching that TLD. Since MAJESTIC\_INDEX1 table has index on “TLD”, so the below command is taking the least time as user is querying the page using TLD

time curl --data "domain2\_idx1=.in" <http://localhost:8018/search2_idx1>

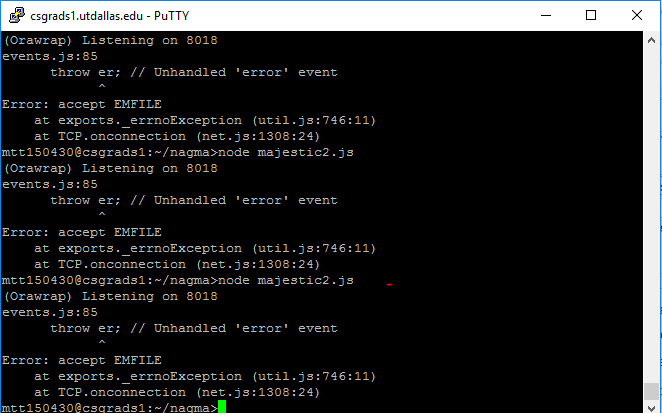
**APACHE JMETER:**

3.3> The desired test plan will be to test the pages developed in Part2. Start with query1 (enter a domain, return its rank). Simulate only 1 user first, then 10, then 50. Run it multiple times? What do you notice? Any errors? Why do you think it is happening? Be specific.

For implementation without using orawrap we are able to push upto 20 users. Errors above 20 concurrent users is shown below:



3.5> With the new implementation using orawrap, we can see 4000 concurrent clients can be pushed through, after which we are getting the error:



**Comment:** Orawrap is a wrapper module for the Oracle Database driver for Node.js ([node-oracledb](https://github.com/oracle/node-oracledb)). Some of the features include:

* A pool manager that provides pool storage and retrieval methods as well as connection request queuing
* A connection manager that provides a simplified execute method (can open and close connections automatically)
* Support for centralizing SQL scripts to be executed at various timings

Therefore using orawrap will allow more concurrent users to access the web server as only one database connection will be opened in this case, thus increasing the performance of the web server. If traditional process is used then for every query ,oracle database connection is opened which decreases the performance.

**3.5> Comparison for both part 2 queries with & without using orawrap:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | No. of concurrent users | |
| Query | Table Hit | Not using orawrap | Using orawrap |
| Query 1 | MAJESTIC | 15 | 4000 |
| MAJESTIC\_INDEX1 | 15 | 4000 |
| MAJESTIC\_INDEX2 | 15 | 4000 |
| MAJESTIC\_INDEX3 | 15 | 4000 |
| Query 2 | MAJESTIC | 15 | 4000 |
| MAJESTIC\_INDEX1 | 15 | 4000 |
| MAJESTIC\_INDEX2 | 15 | 4000 |
| MAJESTIC\_INDEX3 | 15 | 4000 |