

273 Lab1 Report

Jiongfeng Chen ID: 010830105

“Lab 1 : Using REST (Node.js), HTML 5 and Angular JS”

Introduction

Goals and purpose of the system: This is the report for the personal Lab1 of course CmpE273, spring 2016. In this Lab1, I designed and implemented an online Calculator, a Twitter application. They are developed based on Node.js, Express, MySQL and Angular.js. In this report, the system design, system functions and performance will be presented by following the important queries from the lab1 requirement document.

System Design

The system architecture, logical operations and system components are designed based on Node.js platform and Express framework, according to the requirements. The strategies, such as validations, password encryption, session mechanism, and connecting pool, etc. have been implemented to enhance the scalability, performance and reliability of the web application system.

The systems are designed to demonstrate stateless web services (REST). For the database, to express statefull behavior some persistence providing technologies need to be used. MySQL is

the optimum choice. Jmeter was applied to automate the processes of test the performance such as average response time of the web application. The connection pool of MySQL has created for scalable server-side and networking applications. The functions of this twitter application include but not limited to sign up, sign in, sign out, profile, tweets, retweets, follow specific user, unfollow specific user, hashtag, connection pool. The functions of the calculator include addition, subtraction, multiplication, and division.

Results

Part 1 – Calculator (4 points)

Addition

Welcome to calculator

2.1 + 1 calculate 3.1

Subtraction

Welcome to calculator

2 - 1 calculate 1

Multiplication

Welcome to calculator

2.1 * 2 calculate 4.2

Division

Welcome to calculator

/ 2

Exception handle

Welcome to calculator

/ can not be zero

Testing should be done using JMeter: average time for each operation (2 Points). Compare and discuss briefly

Part 2: Twitter Application (16 points)

(4 points)

1. Basic User functionalities:
 - a. Sign up the new users

273_student01@sjsu.edu Password sign in

Email student@sjsu.edu

Full Name Firstname Lastname

Password Password

Repeat Password Repeat Password

Sign up

Twitter Lab1 273_student01@sjsu.edu sign in

Passwords have to be encrypted

```
MySQL 5.7 Command Line Client
mysql> ^C
mysql> use twitter_db;
Database changed
mysql> select * from user;
+----+-----+-----+-----+-----+
| id | email | username | password | create_at |
+----+-----+-----+-----+-----+
| 1 | 273_student01@sjsu.edu | student01 | 123456 | 2016-03-10 10:01:01 |
| 2 | 273_student02@sjsu.edu | student02 | 123456 | 2016-03-11 10:01:01 |
| 3 | 273_student03@sjsu.edu | student03 | 123456 | 2016-03-13 10:01:01 |
| 4 | 273_student04@sjsu.edu | 12 | c20ad4d76fe97759aa27a0c99bffa6710 | 2016-02-18 23:06:35 |
| 5 | 273_student05@sjsu.edu | 1 | c4ca4238a0b923820dcc509a6f75849b | 2016-02-19 02:23:23 |
+----+-----+-----+-----+-----+
5 rows in set (0.03 sec)

mysql>
```

Logout

Post

All tweets

student01

my 1st tweet, yeah.

2016-02-20T04:59:13.000Z

retweet

reply

0 K/s

0 K/s

c. Sign out

Logout

Post

Tweet

Logout

my tweet

Post

All tweets

student01

my 1st tweet, yeah.

2016-02-20T04:59:13.000Z

retweet

reply

student01

my 2nd tweet.

Retweet

Retweet

×

retweet

my 1st tweet, yeah.

close

submit

Post

reply

reply

2016-02-20T11:02:09.000Z

retweet

reply

Follow

Logout

student01

All tweets

student01

my 1st tweet, yeah.

student01

my 2nd tweet.

student01

my 3rd tweet...

Logout

student02

Follow

All tweets

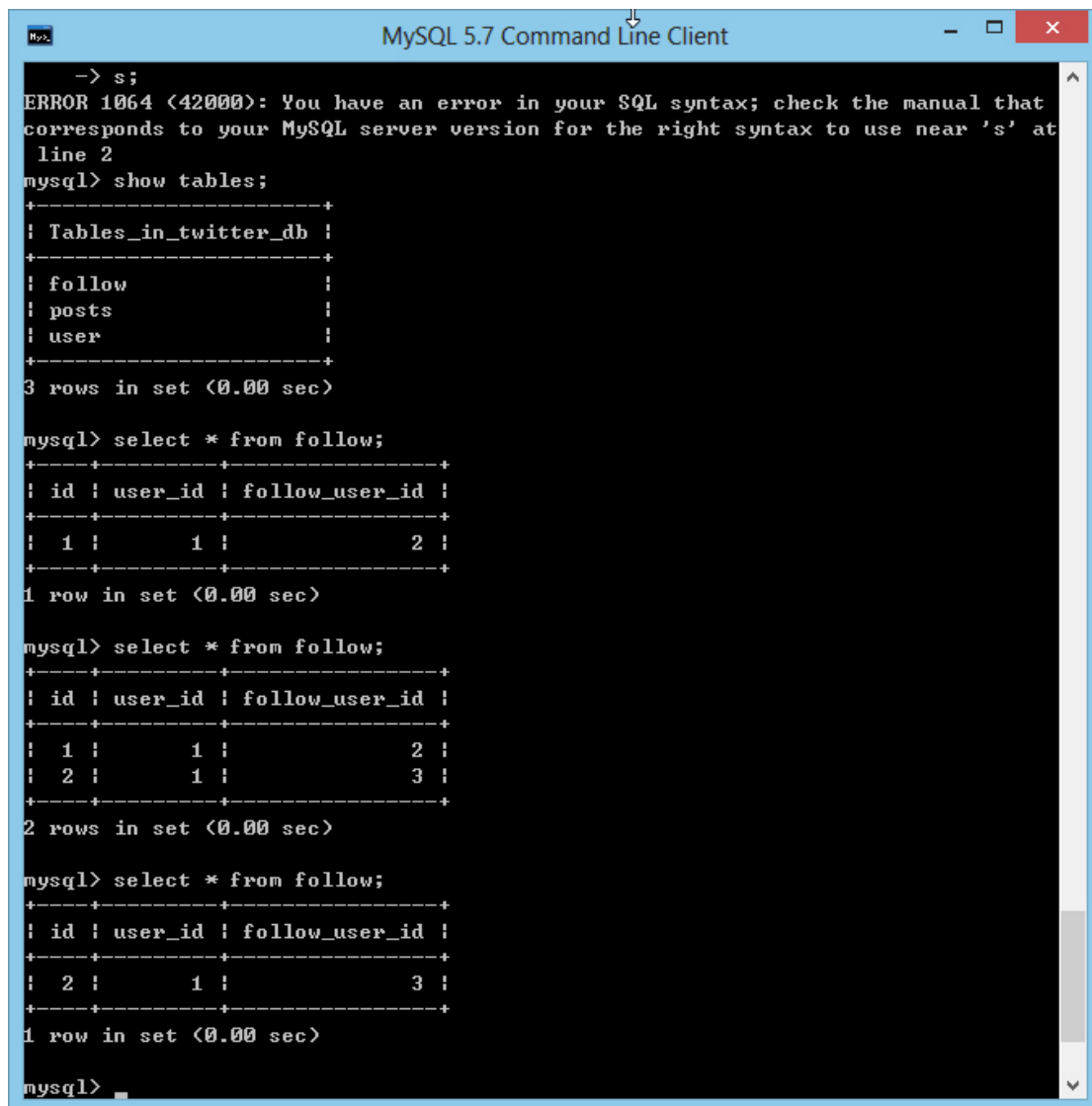
After follow, button changed to unfollow

Logout

student02

Unfollow

All tweets



```
MySQL 5.7 Command Line Client

-> s;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 's' at
line 2
mysql> show tables;
+-----+
| Tables_in_twitter_db |
+-----+
| follow                |
| posts                 |
| user                  |
+-----+
3 rows in set (0.00 sec)

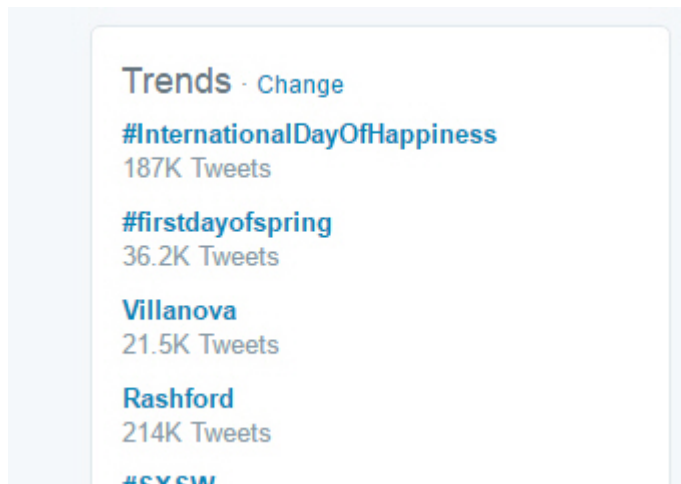
mysql> select * from follow;
+----+-----+-----+
| id | user_id | follow_user_id |
+----+-----+-----+
| 1  | 1       | 2              |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from follow;
+----+-----+-----+
| id | user_id | follow_user_id |
+----+-----+-----+
| 1  | 1       | 2              |
| 2  | 1       | 3              |
+----+-----+-----+
2 rows in set (0.00 sec)

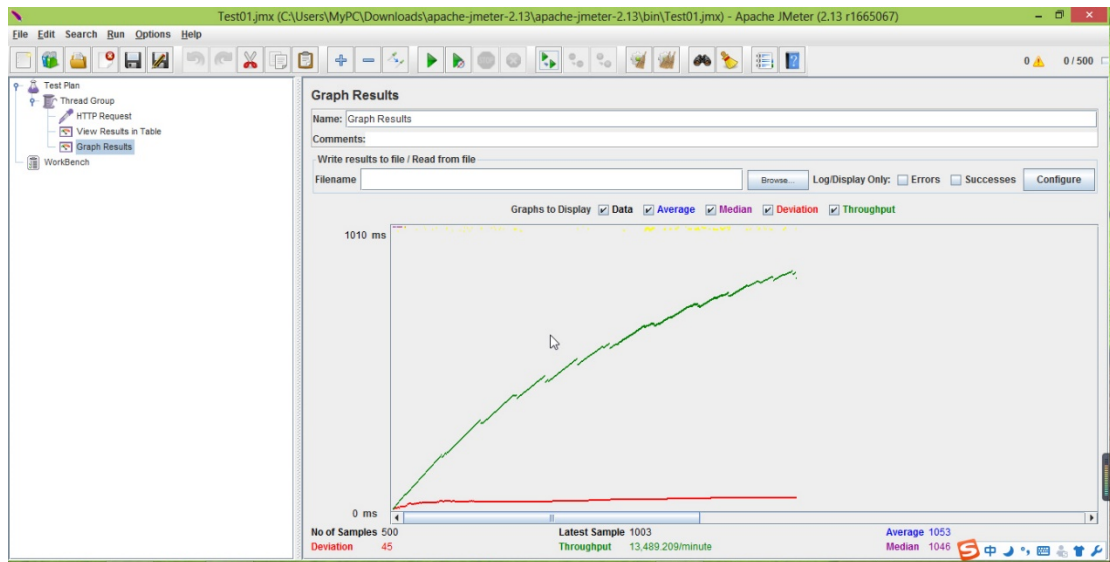
mysql> select * from follow;
+----+-----+-----+
| id | user_id | follow_user_id |
+----+-----+-----+
| 2  | 1       | 3              |
+----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

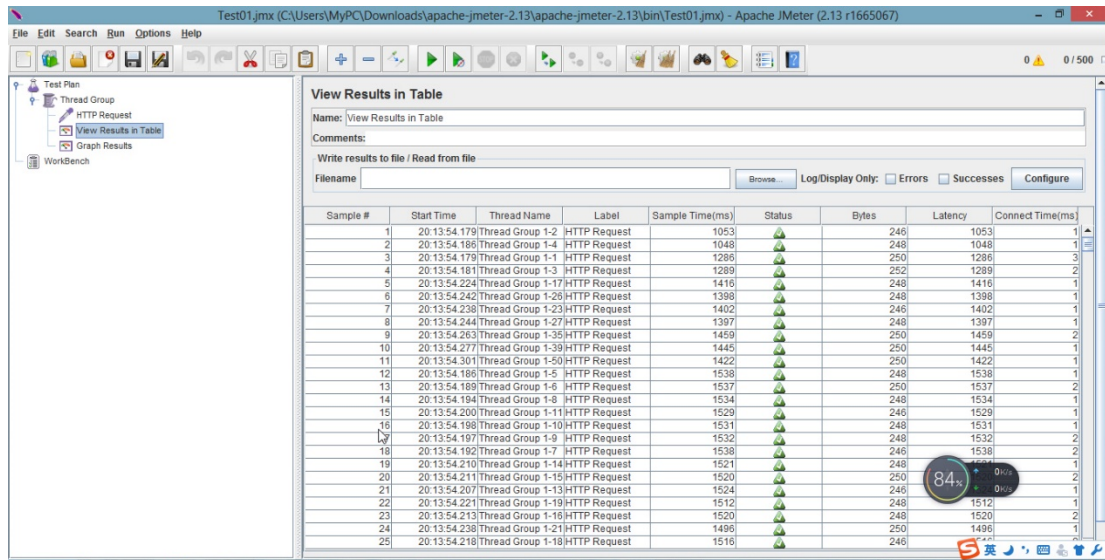
Hashtag



Connection pool with 500



Without Connection pool



Part 3: Answers to Questions

1 The encryption algorithm used in your application.

Cryptographic hash function, MD5 has been used in the application. There are several popular encryption algorithms, such as MD5, SHA, AES, PGP, ADS et al. The MD5 hashing is a very good way to store passwords. Compared to others, it is easy to use and the hashes are inherently one-way in nature. By storing passwords in hash format, it's very difficult for someone with access to the raw data to reverse it. While AES is applied in it comes to symmetric key encryption. PGP is the most popular public key encryption algorithm. The encrypted strings can be reversed back into their original decrypted form if you have the right key.

2. Compare the Results of the graphs with and without connection pooling.

Connection pool is more efficient, and it also save the valuable resources of memory and I/O band. It performed one time connection only, and thus avoid repeating connection which will dramatically decrease the performance. In our testing results, as the figure shows, the performance are 30 percent better than the one no connection pool, in the cases of 300, 500 concurrent user.

3.How would you implement Request Caching?

The solution would be deploying proxy to process the requests. It can be several levels accordingly. The handshake cache would be used by applying proxy to get the benefit. For instance, a client end forward proxy at the client end and a server end reverse proxy would be deployed. When the request to the forward proxy is sent, the server does a cache lookup. If miss the cache, the forward proxy digests the body and sends only the digest to the reverse proxy. Then, the reverse server will look for a match in the request cache and, if found, sends that request to the server.