SYSTEM DESIGN

The whole program mainly consists of two parts: registry module and communication module. All the system is kept on both server and client end.

The program starts with the registry service. The registry service firstly starts on server end, which provides binding method and responds to client-end registry service. ServerRegistry has already been optimized for multi-user supporting. The service begins when ServerRegistry is instantiated. It provides a bind method to bind a string name with the service the server can provide. The binding information is kept in server registry service as a ConcurrentHashMap. Three different ConcurrentHashMap kept different relationships of data. At the same time, it provides the concurrent accessing and editing support suit for large concurrent situation. The service starts listening the clients' request at the time it was instantiated in another thread. Once it receives a request, it will return different services such as look-up and invoke. The result will then be returned to clients by the communication module.

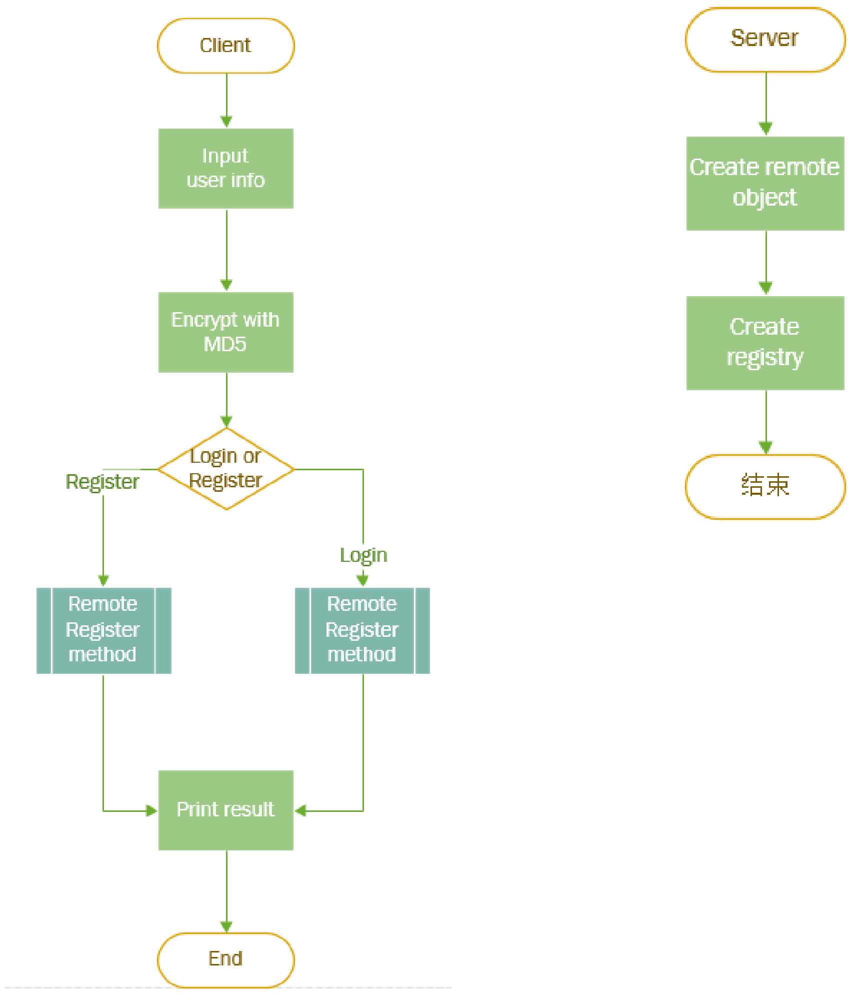
The registry service on client end is simpler. It only requests the server's host address and socket number to be instantiated. Once it is operating, it first needs to find a specific service operating on the remote server, so it needs a look-up operation. The look-up operation will send the server the service name it is looking for. If the server holds this service at that time, the server will then instantiate a proxy object, we call it stub here. This stub will be returned to client. The client can use this stub to invoke the methods contained in the interface as invoking the local methods. At the same time, the stub will collect the invoking information such as the invoked method name, Object key and the invoking parameters, pack them into a TranSegment class (used particularly for packing these information) and then transfer to the server. For the client end, the invocation process has no difference from invoking a local method.

The communication module provides methods of passing different kinds of messages between server and clients. In this module, I implemented two methods, SendObj and RecObj. As they are named, they are used to send and receive object. The sending and receiving of object here implemented the serialization methods provided in Java API, which can transfer the serialized object in Java serialization form.

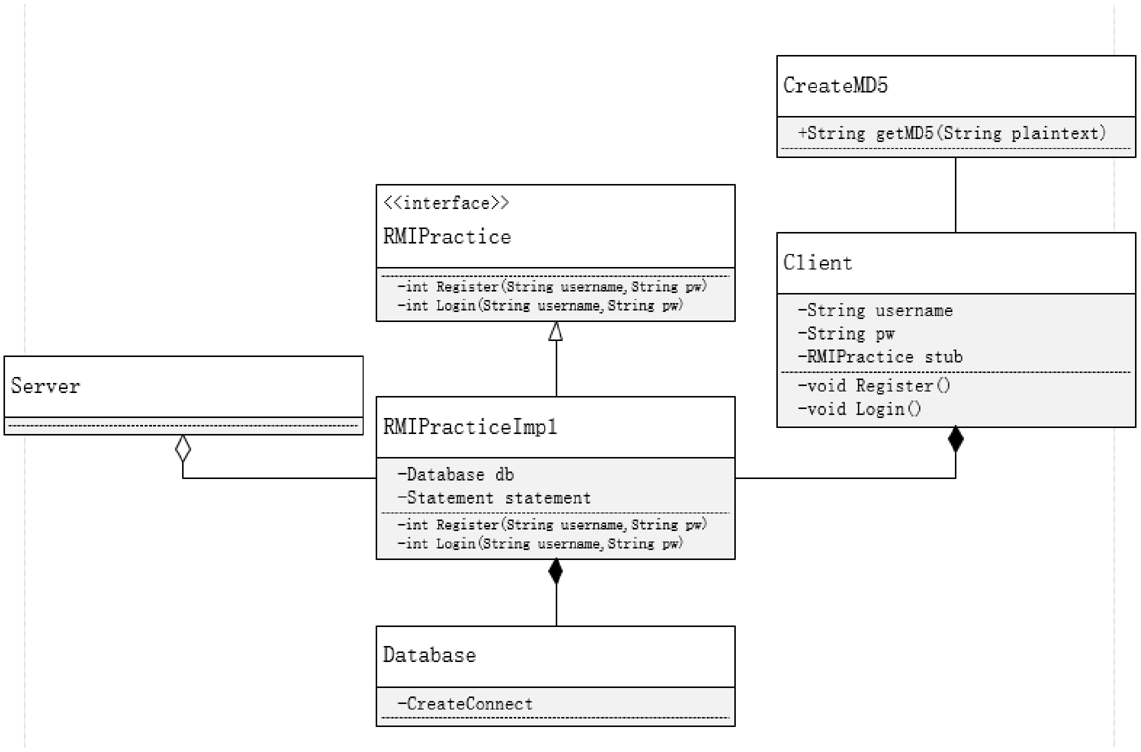
After all these basic implementations, the program is optimized for high-concurrency supporting. A thread pool is used to handle threads. At the very beginning, the RegistryServer start a thread for listening the connections. At the same time, it can process the binding request, which means that it can load the service dynamically. The connected clients' request will be processed in isolated threads.

TESTING: REGISTER & LOGIN SERVICE

Similar to the use of java rmi. The flow chart of the testing module is shown below.



The class diagram is attached below.


CreateMD5
+String getMD5(String plaintext)
<<interface>>
RMIPractice
-int Register(String username, String pw)
-int Login(String username, String pw)
Client
-String username
-String pw
-RMIPractice stub
-void Register()
-void Login()
Server
RMIPracticeImpl
-Database db
-Statement statement
-int Register(String username, String pw)
-int Login(String username, String pw)
Database
-CreateConnect

This test system includes an interface which will be held in both server and client ends. The class that implements the interface will only be stored on server end. This implementation uses MySQL to store users' information. To instantiate the MySQL connection, modify the Server class where instantiated RMIPracticeImpl class. Use the specified database name, user name and password to initialize connection.

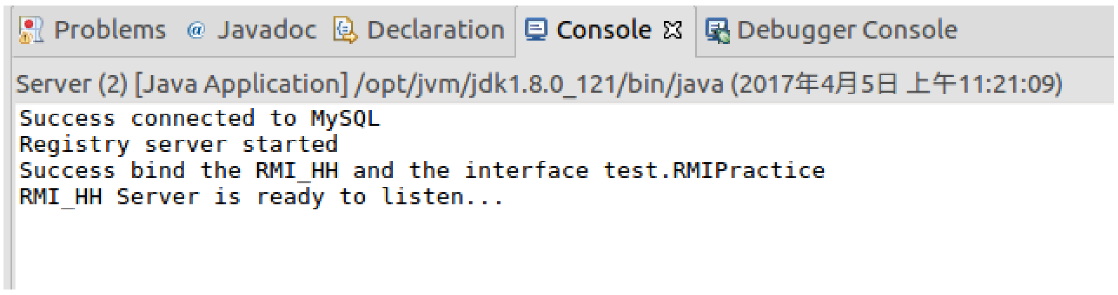
Furthermore, a ConcurrentClient class is added to the testing system, which can generate a number of concurrent register request to server, specified by the int value in it. This class is used to test the concurrency of the system.

To start the testing system, operate the server class first, then operate the client or ConcurrentClient class. The server and the client will print out the status of the process. You can also check the MySQL to examine the completeness.

RUNNING RESULT

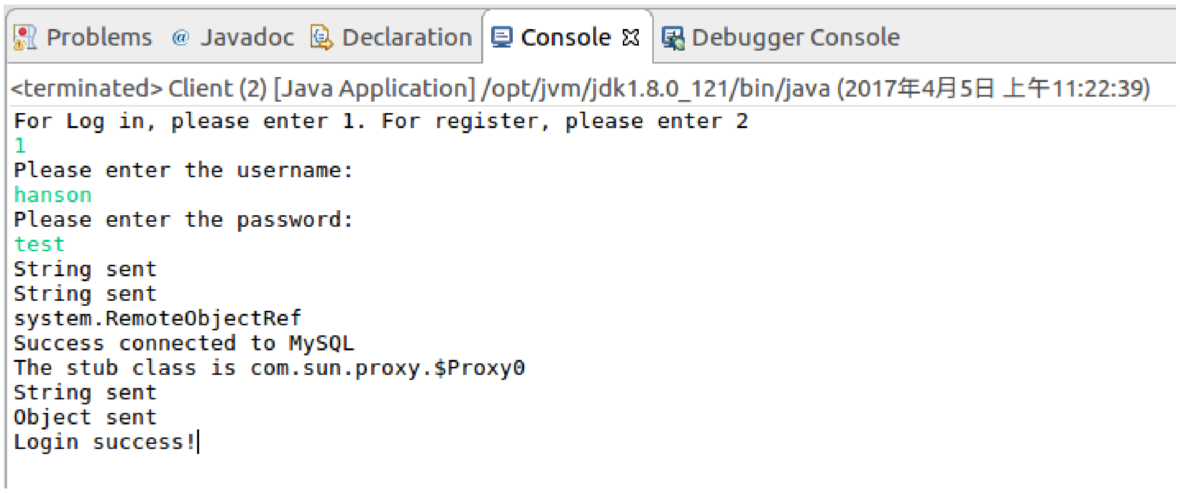
The test of the program uses the last Lab assignment, with just two lines of correction when instantiating the registry service.

1. Start the server



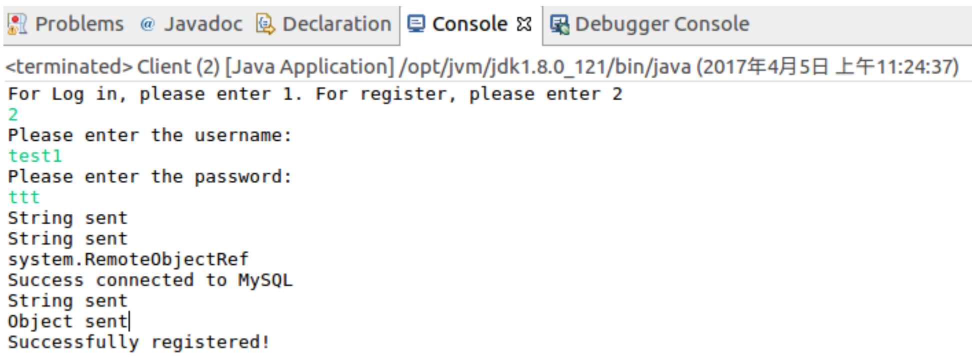
```
Server (2) [Java Application] /opt/jvm/jdk1.8.0_121/bin/java (2017年4月5日 上午11:21:09)
Success connected to MySQL
Registry server started
Success bind the RMI_HH and the interface test.RMIPractice
RMI_HH Server is ready to listen...
```

2. Start the client and the login process



```
<terminated> Client (2) [Java Application] /opt/jvm/jdk1.8.0_121/bin/java (2017年4月5日 上午11:22:39)
For Log in, please enter 1. For register, please enter 2
1
Please enter the username:
hanson
Please enter the password:
test
String sent
String sent
system.RemoteObjectRef
Success connected to MySQL
The stub class is com.sun.proxy.$Proxy0
String sent
Object sent
Login success!
```

3. Register process

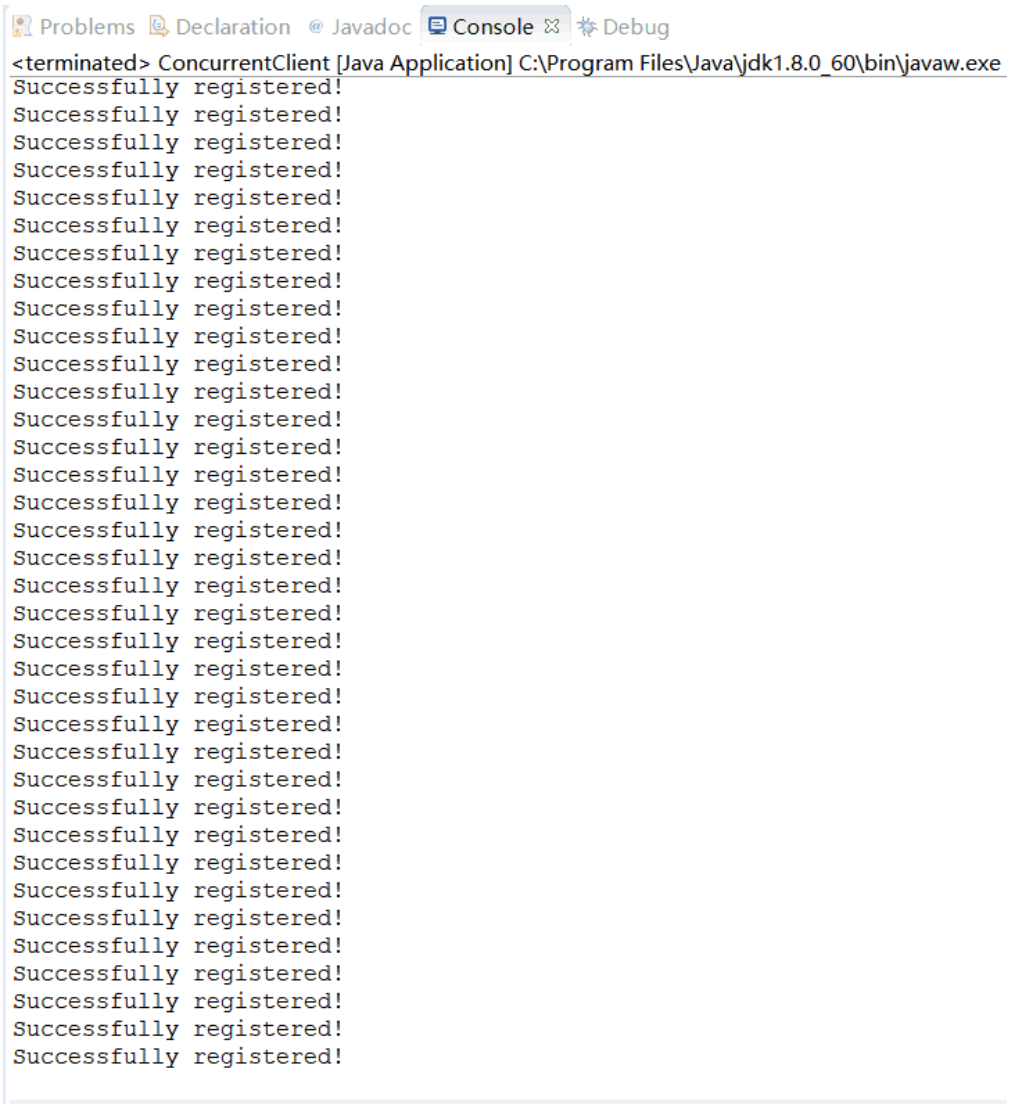


```
<terminated> Client (2) [Java Application] /opt/jvm/jdk1.8.0_121/bin/java (2017年4月5日 上午11:24:37)
For Log in, please enter 1. For register, please enter 2
2
Please enter the username:
test1
Please enter the password:
ttt
String sent
String sent
system.RemoteObjectRef
Success connected to MySQL
String sent
Object sent
Successfully registered!
```

4. Concurrency testing

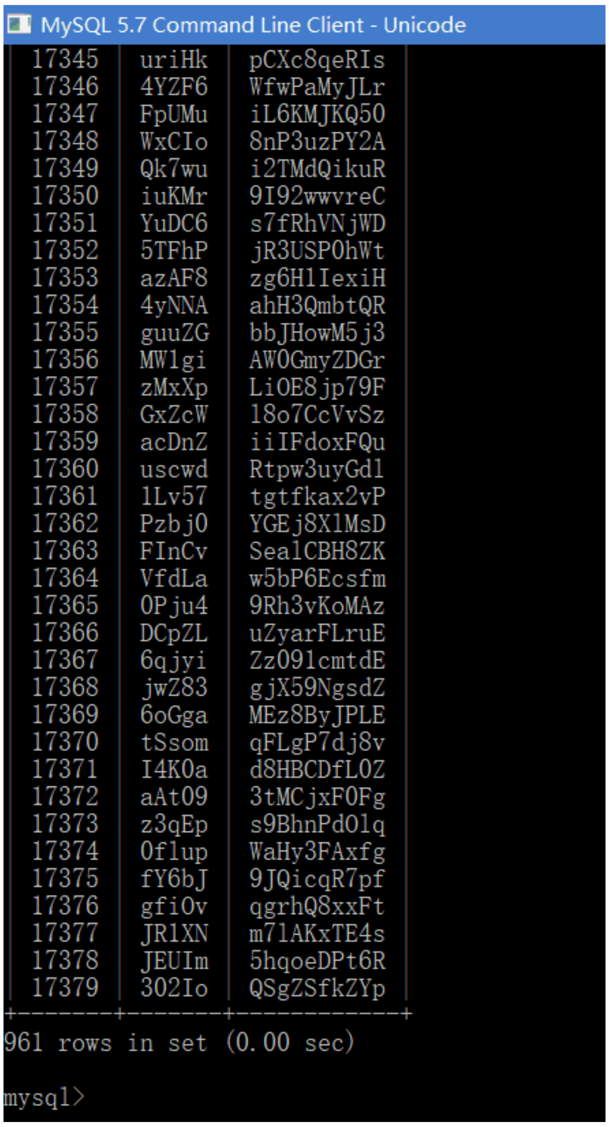
Concurrent_num = 1000

Eclipse console (can only display limited number of commands):



```
<terminated> ConcurrentClient [Java Application] C:\Program Files\Java\jdk1.8.0_60\bin\javaw.exe
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
Successfully registered!
```

Result in Database:



MySQL 5.7 Command Line Client - Unicode

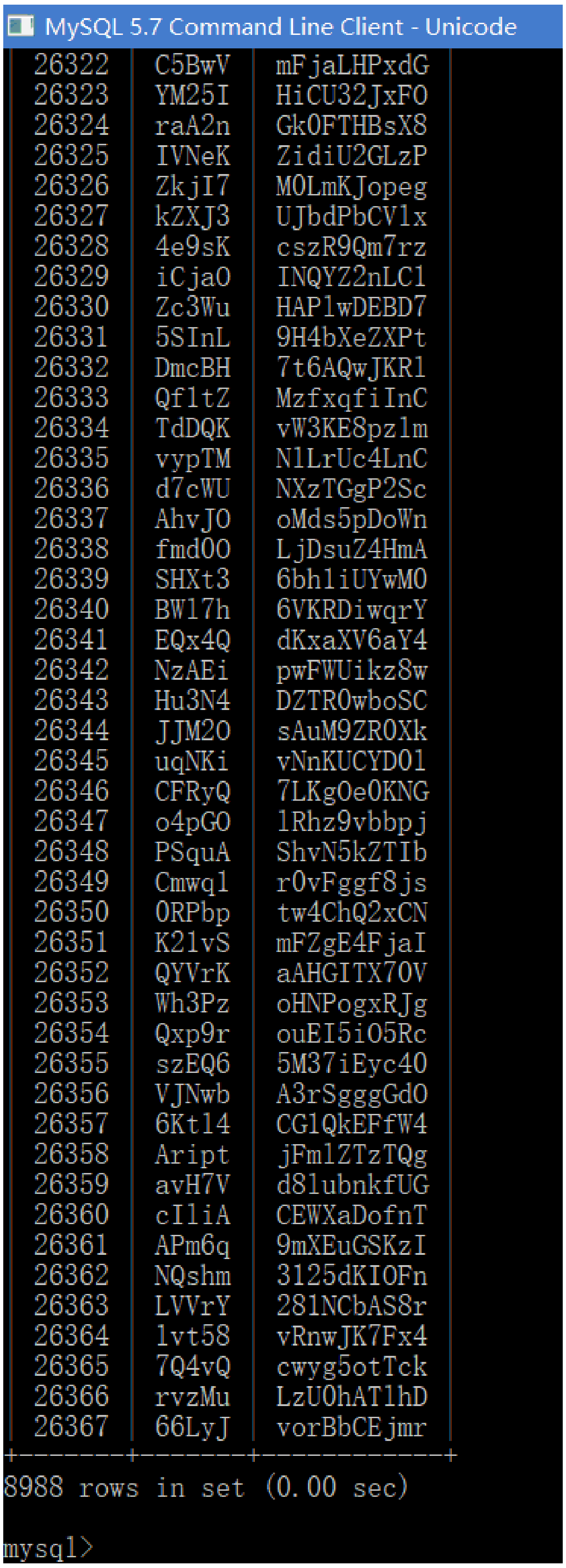
```
| 17345 | uriHk  | pCXc8qeRIs  |
| 17346 | 4YZF6  | WfwPaMyJLr  |
| 17347 | FpUMu  | iL6KMJKQ50  |
| 17348 | WxCIo  | 8nP3uzPY2A  |
| 17349 | Qk7wu  | i2TMdQikuR  |
| 17350 | iuKMr  | 9I92wwvreC  |
| 17351 | YuDC6  | s7fRhVNjWD  |
| 17352 | 5TFhP  | jR3USP0hWt  |
| 17353 | azAF8  | zg6H1IexiH  |
| 17354 | 4yNNA  | ahH3QmbtQR  |
| 17355 | guuZG  | bbJHowM5j3  |
| 17356 | MW1gi  | AW0GmyZDGr  |
| 17357 | zMxXp  | LiOE8jp79F  |
| 17358 | GxZcW  | 18o7CcVvSz  |
| 17359 | acDnZ  | iiIFdoxFQu  |
| 17360 | uscwd  | Rtpw3uyGdl  |
| 17361 | 1Lv57  | tgtfkax2vP  |
| 17362 | Pzbj0  | YGEj8X1MsD  |
| 17363 | FInCv  | Sea1CBH8ZK  |
| 17364 | VfdLa  | w5bP6Ecsfm  |
| 17365 | 0Pju4  | 9Rh3vKoMAz  |
| 17366 | DCpZL  | uZyarFLruE  |
| 17367 | 6qjyi  | Zz091cmtdE  |
| 17368 | jwZ83  | gjX59NgsdZ  |
| 17369 | 6oGga  | MEz8ByJPLE  |
| 17370 | tSsom  | qFLgP7dj8v  |
| 17371 | I4K0a  | d8HBCDfL0Z  |
| 17372 | aAt09  | 3tMCjxF0Fg  |
| 17373 | z3qEp  | s9BhnPd01q  |
| 17374 | 0f1up  | WaHy3FAxfg  |
| 17375 | fY6bJ  | 9JQicqR7pf  |
| 17376 | gfi0v  | qgrhQ8xxFt  |
| 17377 | JR1XN  | m71AKxTE4s  |
| 17378 | JEUIm  | 5hqoeDPt6R  |
| 17379 | 302Io  | QSgZSfkZYp  |
+-------+--------+-------------+
961 rows in set (0.00 sec)

mysql>
```

961 registers are succeeded, while the rest of the requests failed.

Concurrent_num = 10000

Result in Database:



```
| 26322 | C5BwV | mFjaLHPxdG |
| 26323 | YM25I | HiCU32JxFO |
| 26324 | raA2n | Gk0FTHBsX8 |
| 26325 | IVNeK | ZidiU2GLzP |
| 26326 | ZkjI7 | M0LmKJopeg |
| 26327 | kZXJ3 | UJbdPbCV1x |
| 26328 | 4e9sK | cszR9Qm7rz |
| 26329 | iCja0 | INQYZ2nLC1 |
| 26330 | Zc3Wu | HAP1wDEBD7 |
| 26331 | 5SInL | 9H4bXeZXPt |
| 26332 | DmcBH | 7t6AQwJKR1 |
| 26333 | Qf1tZ | MzfxqfiInC |
| 26334 | TdDQK | vW3KE8pz1m |
| 26335 | vypTM | N1LrUc4LnC |
| 26336 | d7cWU | NXzTGgP2Sc |
| 26337 | AhvJO | oMds5pDoWn |
| 26338 | fmd0O | LjDsuZ4HmA |
| 26339 | SHXt3 | 6bh1iUYwMO |
| 26340 | BW17h | 6VKRDiwqrY |
| 26341 | EQx4Q | dKxaXV6aY4 |
| 26342 | NzAEi | pwFWUikz8w |
| 26343 | Hu3N4 | DZTR0wboSC |
| 26344 | JJM2O | sAuM9ZR0Xk |
| 26345 | uqNKi | vNnKUCYD01 |
| 26346 | CFRyQ | 7LKg0e0KNG |
| 26347 | o4pGO | 1Rhz9vbbpj |
| 26348 | PSquA | ShvN5kZTIb |
| 26349 | Cmwq1 | r0vFggf8js |
| 26350 | ORPbp | tw4ChQ2xCN |
| 26351 | K21vS | mFZgE4FjaI |
| 26352 | QYVrK | aAHGITX70V |
| 26353 | Wh3Pz | oHNPogxRJg |
| 26354 | Qxp9r | ouEI5iO5Rc |
| 26355 | szEQ6 | 5M37iEyc40 |
| 26356 | VJNwb | A3rSgggGdO |
| 26357 | 6Kt14 | CG1QkEFfW4 |
| 26358 | Aript | jFm1ZTzTQg |
| 26359 | avH7V | d81ubnkfUG |
| 26360 | cIliA | CEWXaDofnT |
| 26361 | APm6q | 9mXEuGSKzI |
| 26362 | NQshm | 3125dKIOFn |
| 26363 | LVVrY | 281NCbAS8r |
| 26364 | 1vt58 | vRnwJK7Fx4 |
| 26365 | 7Q4vQ | cwyg5otTck |
| 26366 | rvzMu | LzU0hAT1hD |
| 26367 | 66LyJ | vorBbCEjmr |
+-------+-------+------------+
8988 rows in set (0.00 sec)

mysql>
```

8988 registers are succeeded, while the rest of the requests failed.