```
 ____    ___     ___     ___
|___ \  / _ \   / _ \   / _ \
  __) || | | | | (_) | | (_) |
 |__ < | | | |  \__, |  > _ <
 ___) || |_| |    / /  | (_) |
|____/  \___/    /_/    \___/
```

DataPull Documentation

Email JoshLat40@gmail.com for further questions or help

Contents

1.)Required “Tools”

1-1.)Hardware

First things first, we use ethernet->USB adapters because our tablets do not have ethernet ports. The adapters are also essential as they allow us to statically assign an IP to the computer to be used when the ethernet adapter is connected. Each tablet has an IP formatted as 10.30.98.XX where as XX being numbers 1->15, for example I use my master database tablet on IP 10.30.98.1
Yes we could have had all the computers use the same IP as long as it’s different from the master tablet but we made them seperate to give us the ability to (when supported by the DataPuller) use a USB hub and connect to all tablets at once and rip data from all 7 scouting tablets all in one click of a button.

1-2.)Software

Now on to the juicy stuff, let's start with the database (this will be long and confusing so read slowly). We created a MySQL database using MySQL Workbench and embedded multiple tables inside the database each setup to store information from the scouting tablets in a very understandable format. We've created a user account for the database that has special permissions to each table, which only grant the account permission to execute, read, and write to the tables. This special account is utilized by the DataPull program to allow it to gather the data it needs from the client.

2.)How It Works

2-1.)Data Pull

There are multiple stages the program uses in order to extract the data from the client.

2-1-1.)A problem i encountered was the inability to set a timeout for when the master could not connect to the clients SQL database. To overcome this we first ping the client PC’s IP with a timeout of 1ms, if the ping is successful it continues to connect to the client’s SQL database instantly.

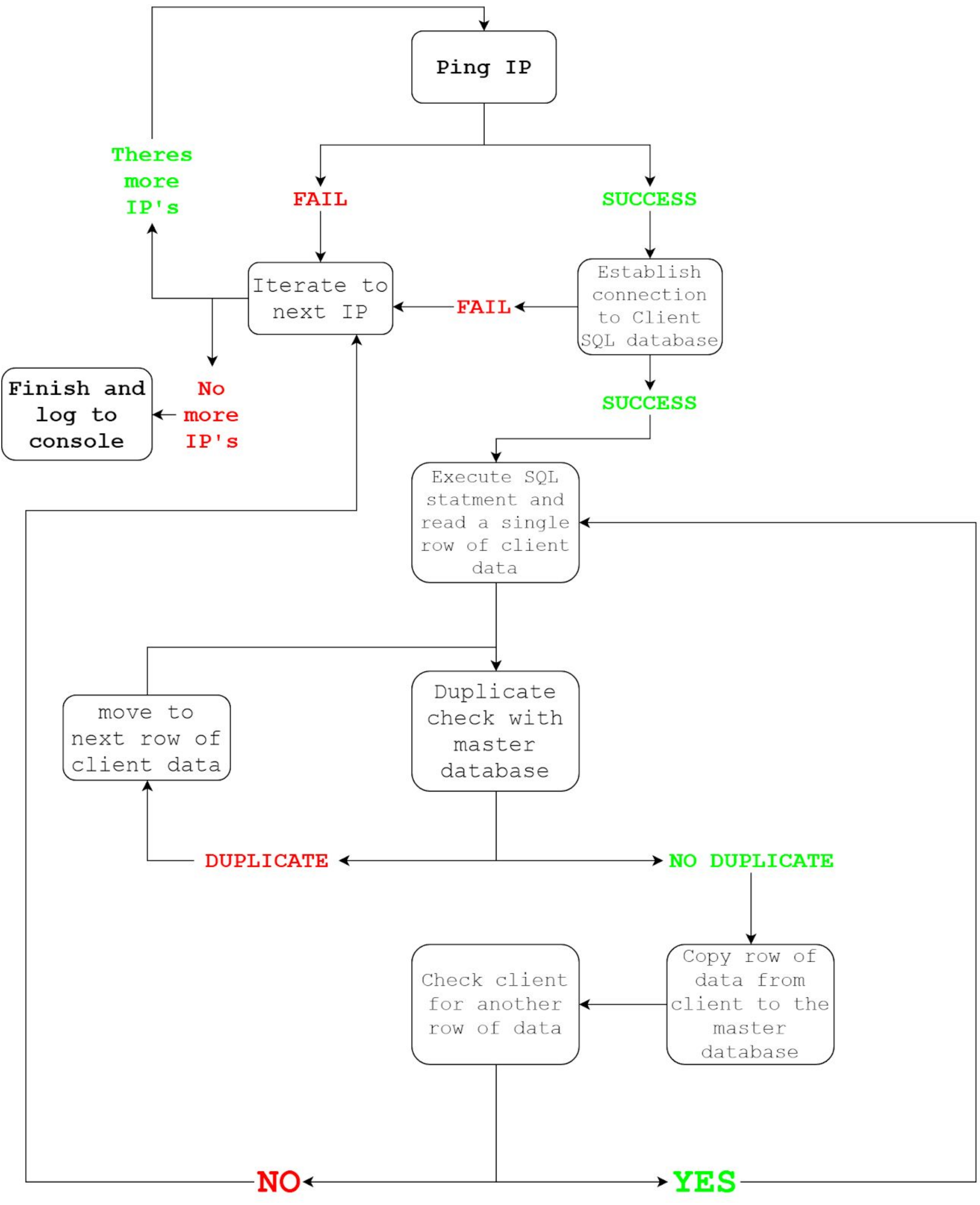
2-1-2.) Now that we’re connected to the clients SQL database and logged in, we can now execute a prepared statement which will return all of the data from the table we chose.

2-1-3.) With this data two things happen back and forth to work together to transfer the data from client to the master. We first grab the first row of data,then using a key (a timestamp in the data) we run a statement on the master database, iterate through each row of data and compare the key from the clients data to the master’s data, if there is not a match then the program creates a statement to insert to data to the master database; however if there is a match then that row of data is ignored and we move on and do the same process with the next row of data from the client database.

2-2.)Data Push

This system is pretty similar to the **Data Pull** method except it is used to push the match schedule from Master->Client, It connects to the client, executes an SQL statement to clear the client's current match schedule, then iterates through each data row from the masters schedule, creates an insert statement and executes the statement then repeats for every row of data in that table.

2-3.) Flow-Chart


Ping IP
FAIL
SUCCESS
Theres more IP's
Iterate to next IP
FAIL
Establish connection to Client SQL database
Finish and log to console
No more IP's
SUCCESS
Execute SQL statment and read a single row of client data
move to next row of client data
Duplicate check with master database
DUPLICATE
NO DUPLICATE
Check client for another row of data
Copy row of data from client to the master database
NO
YES

3.)PHP Methods Utilized

3-1.)console($text)

Logs back to the page $text

3-2.)ping($host, $port)

Pings $host using $port with a preset timeout of 1ms returning a boolean value

3-3.)ExecuteSQLGET($tbl, $sql)

Executes a SQL statement on the Master database on $tbl, the statement being executed is $sql

3-4.)ExecuteSQLDELPOY($tbl, $sql)

Executes a SQL statement on the Client database on $tbl, the statement being executed is $sql

3-5.)CheckLocalDataDuplicates($tbl, $field)

Checks the Master database $tbl for a row of data with a key matching $field ($field is the key from the Client data), this method keeps the old rows on the client database and only copies new ones to the Master database.

3-6.)HandleDataRIP($tbl, $idcol)

Iterates through the CLients database data individually by row on $tbl ($idcol is the special "key" used in duplicate checking), after duplicate checking returns true, an INSERT statement is formed using the data from that row then executed on the Master database.

3-7.)HandleDataDEPLOY($tbl)

Clears the Clients $tbl, then iterates through the Masters database data individually by row on $tbl, an INSERT statement is formed using the data from that row then executed on the Client database.

3-8.)MainSQL($tbl, $type)

Using $type to determine to Push a match schedule or Pull data, If $type is Push then run **HandleDataPUSH()** If $type is Pull then run **HandleDataRIP()**

3-9.)__construct($tblname)

When called, begins to ping IP's until a connection is established, then connects and logs into the Clients SQL database, finally it runs MainSQL() on the desired $tbl and the option to Push/Pull data is passed with $type. $tblname is the desired datatable to Push/Pull data from/to.