Programming Languages Project: Final Report

Manuel Yafté Gallegos López

05-29-2020

# Concurrent Database Sharing

## Introduction

Nowadays, the world is digital around 4.57 million people use the internet[1], technology is everywhere. This leads to an increasing amount of systems and apps to solve problems or make life easier. Usually the computational systems need to store data to work well and keep records, so with the time organizations can take better decisions based on their historical data.

Software engineering projects have different stages such as analysis, design, implementation, verification and testing[2]. In the early implementation is very common that developers does not have a hosting to allocate  the database that will be required, therefore, each developer, uses its "*localhost*", in other words, its own computer to store provisional data in order to develop and test system functionalities. Most of the times, the project is developed in teams. As mention before, it is very to test functionalities and pair reviews.

## Problematic

When this point is reached, a little inconvenient comes up, the databases are different in every computer, unless you transfer files through some media, which sometimes is boring to look for the right file, download it and the import it.

Some web frameworks like Django offer what is called *migrations.* These, allow to keep tracking of database versions models[3], and solves this little problem somehow because once they are run, they are store in the project, and just when another team member execute them in its computer , it will have same version of database. However, migrations only keep track of models, but not data.

## Proposed solution

To improve the development experience in the early stages, here is presented a tool that provide the ability to concurrently update the database from computer to the other ones, this means, it will be possible to have the same database with the same data in all teams computers.  The program will update database file every certain amount of time, and serving the other computers with the updates.

## Architecture

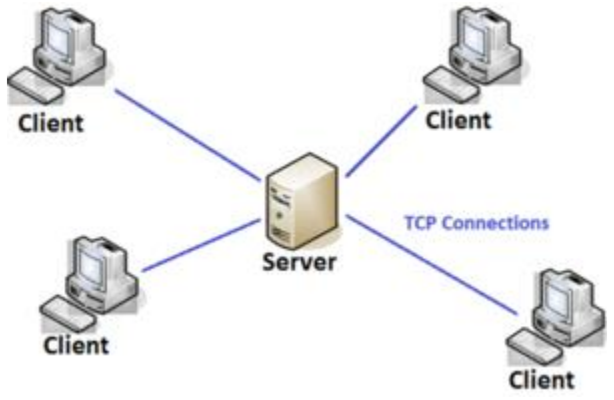


Figure 1. Client-server architecture

To model the solution, a client-server architecture was followed. In this case, the server is the computer who is going to share the database. This architecture consists of one server, which serves to n computers connected to it. They are connected through TCP connections. Java Sockets were used to made this connections between the server and clients (PC's). Sockets provide TCP communication mechanism between two computers [4], what it means a way to communicate asynchronously. Another reason to use sockets is that in this way we can communicate different programs running in different computers and they contain synchronized methods for performing some tasks in such a way that we are able to avoid race conditions or consistency errors in memory. These problems arise when multiple programs (threads) access the same resource without waiting each other to terminate its task.

In conjunction with sockets, threads are included architecture to achieve a concurrent application. This program executes a variety of subprograms instead of just using one as usual. This gives the possibility to run programs almost at the same time, the difference can be so minimal that it can said that they are executing at the same time for the user sought. With this approach we can generate a new thread for handling communication with each client.

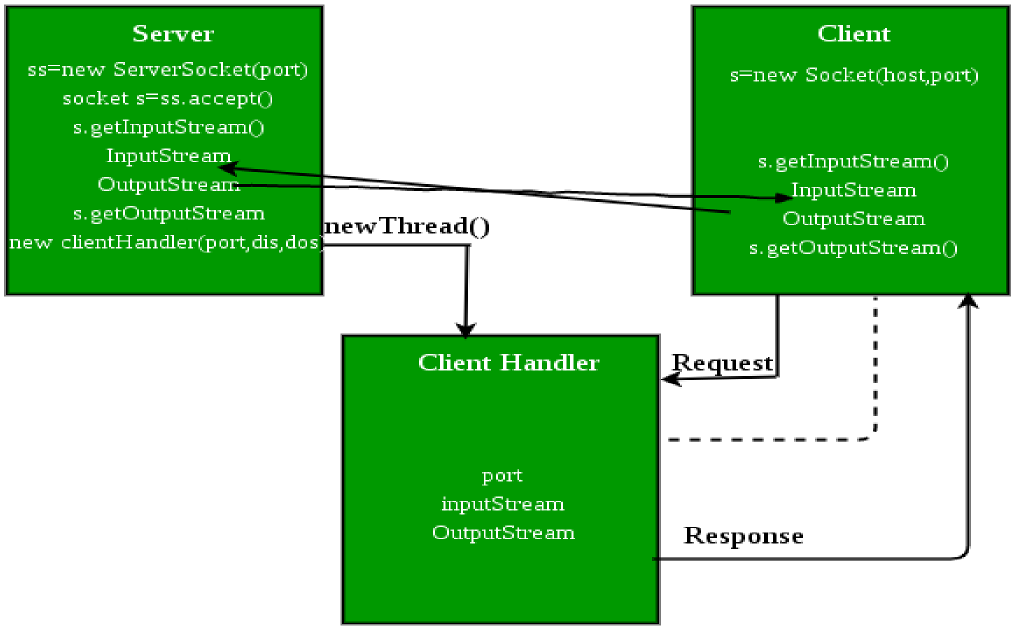


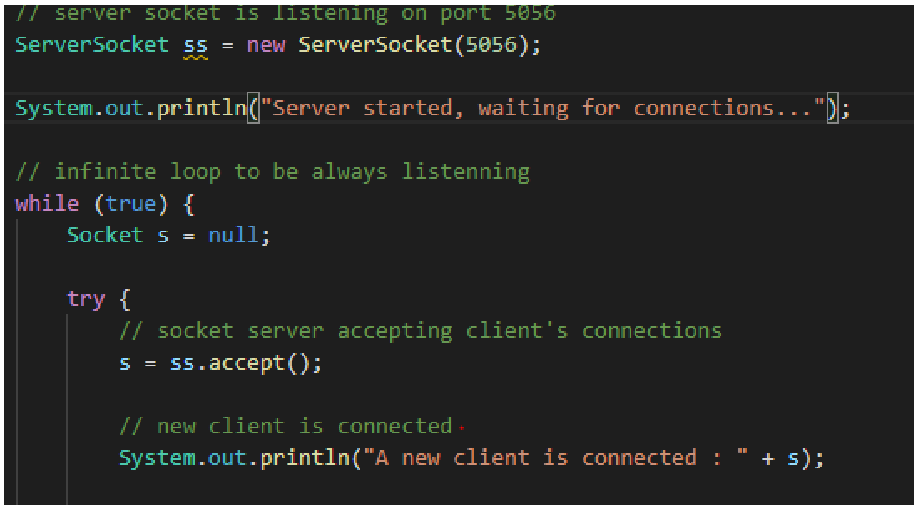
Figure2. Socket and thread working together[5]

As shown in figure 2, there is a class in the middle of Server and Client: ClientHandler has as its responsibilities to create a new Thread per each client that connects to the server. In this way, it is guaranteed that each Client has its own stream to write and read.

## How does it work?

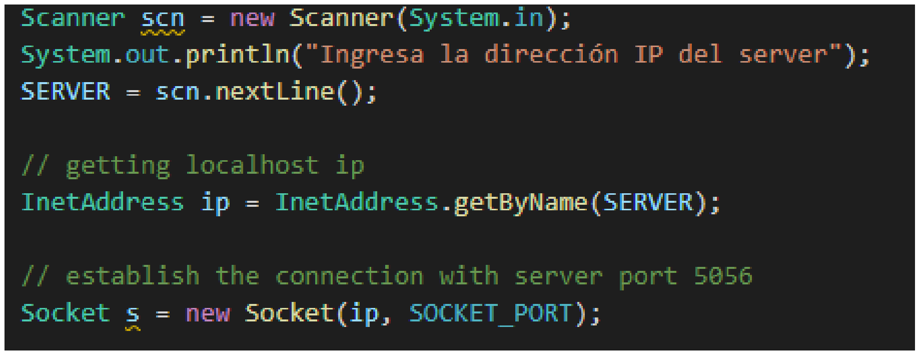
The algorithm goes like this:

- Server Socket is created
- Server socket goes into listening to client’s connection



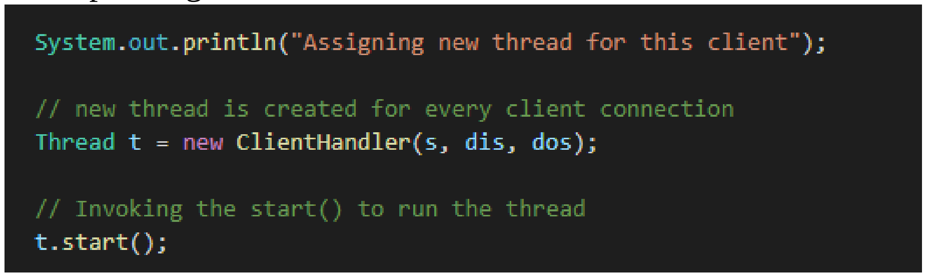
```java
// server socket is listening on port 5056
ServerSocket ss = new ServerSocket(5056);

System.out.println("Server started, waiting for connections...");

// infinite loop to be always listenning
while (true) {
    Socket s = null;

    try {
        // socket server accepting client's connections
        s = ss.accept();

        // new client is connected
        System.out.println("A new client is connected : " + s);
```

- A client is created
  - A client is created
  - Client socket connects to the server socket



```java
Scanner scn = new Scanner(System.in);
System.out.println("Ingresa la dirección IP del server");
SERVER = scn.nextLine();

// getting localhost ip
InetAddress ip = InetAddress.getByName(SERVER);

// establish the connection with server port 5056
Socket s = new Socket(ip, SOCKET_PORT);
```

- On the server side:
  - A new thread is created to handle the communication with each client
  - Database is downloaded from server localhost and write into the corresponding socket stream

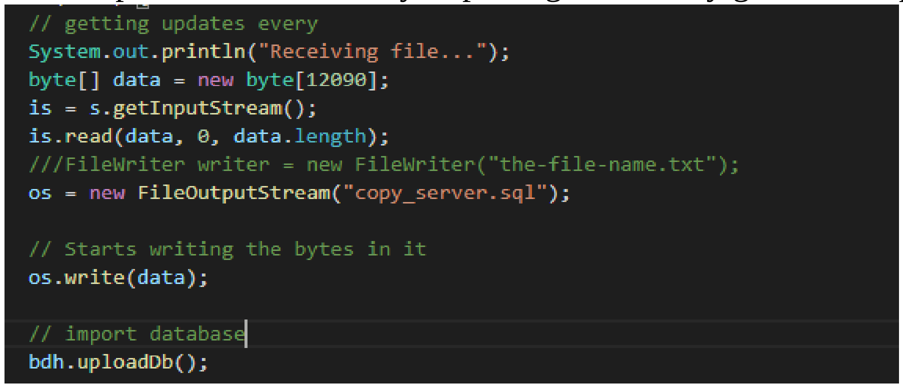


```java
System.out.println("Assigning new thread for this client");

// new thread is created for every client connection
Thread t = new ClientHandler(s, dis, dos);

// Invoking the start() to run the thread
t.start();
```

- Server export its current database and write it to the stream

  - Client reads what was written in the stream
  - Client write bytes in a sql file

- Client updates its database by importing the already generated sql file



```
// getting updates every
System.out.println("Receiving file...");
byte[] data = new byte[12090];
is = s.getInputStream();
is.read(data, 0, data.length);
///FileWriter writer = new FileWriter("the-file-name.txt");
os = new FileOutputStream("copy_server.sql");

// Starts writing the bytes in it
os.write(data);

// import database
bdh.uploadDb();
```

- Repeat the process until user press *“ctrl in”* the console

Results:

- The tool was made to be used in different computers, so I tested with other 2 team members in a development team.
- Results go as expected. The server initialy share its database, but also any updates will be sent to the clients connected every eight seconds. It is possible to add, update and delete registers in a table.
- A video with the application been un in three computers is found here:

https://www.youtube.com/watch?v=JBed5h1y80o

## Conclusions

I believe this project can increase productivity in the development stage where no hosting is provided. As the tool is updating constantly and automatically, developers save time avoiding doing this job manually. I consider this tool could also be used as a generic file transfer with some little adjustments. I want to continue developing this project to make it better and available for more operating systems and come up with possible optimizations. Concurrent programming is powerful and is everywhere, with this project I can release so many things can be done in a multithreading way, which will be faster than using just the main thread.

Programming Languages Project: Final Report

Manuel Yafté Gallegos López

05-29-2020

## Important considerations and prerequisites

- The tool was made to run on different computers, but I you want to run several localhosts on the same computer, you could change the commands in the BdHandler.java file in order to upload the databases in different localhosts.
- This tool was developed for software developers, so it is expected that users have basic knowledge of how to use a command line and feel comfortable with navigate in pc's folders.
- Available only for Windows OS for the moment.
- Available only for mysql databases for the moment. This implies you must have installed xampp to run an Apache server where mysql database runs.
- Clients computers must create their database before start using this tool, it is enough to be created, no need to be filled with any data.

## How to use it?

1. Clone the repository from: https://github.com/Yaf98/programming-languages-project
2. Place files Server.java, Client.java, ClientHandler.java, BdHandler.java and FileManager.java in the bin folder, which should be inside of your disk/xampp/mysql. Example: C:\xampp\mysql\bin
3. Create database in every client if it is missing.
    a. It can be done in the default GUI for mysql: phpMyAdmin or you can do with the command line prompt following the next steps:
        i. Open your sql shell
        ii. Enter to the database you want to modify. If you are going to create it, just type *create database name_database*
2. Run the server in the terminal as a java program.
    a. Run file: java Server
3. Run the client in corresponding computer as java program to connect to server:
    a. Compile file: javac Server.java
    b. Run file: java Server
4. If you are the server, when you update something in your database, changes will be sent to clients
    a. Close connection by pressing ctrl + c in the server terminal
5. If you are a client, you will receive updates very 8 seconds of any changes made.
    a. Close connection by pressing ctrl + c in the client terminal

Git repository:

https://github.com/Yaf98/programming-languages-project

References:

[1] Accessed on: May.28, 2020 https://datareportal.com/
[2] Accessed on: May.28, 2020 https://medium.com/omarelgabrys-blog/software-engineering-software-[3] process-and-software-process-models-part-2-4a9d06213fdc
[4] Accessed on: May.28, 2020 https://docs.djangoproject.com/en/3.0/topics/migrations/
Accessed on: May.28, 2020 [5] https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html