FINAL PROJECT

Sezer Demir

June 9, 2021

1 Requirements

For that project I achieved all the requirements those are mentioned in project besides one thing that if so many data flow occurs sometimes some threads stuck at recv function if client has more than 1 query request.

2 Design Decisions

a.) All arguments, those are passed to program, are checked. Arguments must be valid. Otherwise an error message will be displayed and program ends immediately.

b.) Non ASCII characters are invalid for my project.

c.) Socket is always opened on IP of "127.0.0.1" by the server.

d.) I chose 3d-array to store csv file. Because it's easier to use and I can just replicate the a data frame with it. Each first dimension the indicates column number and second dimension indicates a field of that column.

e.) I didn't check query or csv file's content is proper or not.

f.) All types of query types works except one situation that if there is '=' sign in query to indicate equality or assigning (for update query) there shouldn't be empty space between column name and '=' sign or field value and '=' sign.

g.) Some valid queries:

  (a) SELECT "column1", 'column2', "'column3'", "'column4'", "col umn" FROM TABLE;

  (b) UPDATE TABLE SET "column1"="12", 'column2'=12, "'column3'"="1 2", "'column4'"='1 2', "col umn"="SOME SENTENCE" FROM TABLE;

h.) Since there is no information about UPDATE Query Response, server side sends the rows those are changed after query execution to the client.

i.) To achieve singleton design pattern for server side, I create a named semaphore with O_EXCL and O_CREAT flags so if another server process is running meanwhile, new process knows it's exist and terminates itself immediately.

3 Communication Over Socket

How I approach to communication problem is that when a process wants to send data over socket to another one (from server to client or client to server), firstly sender sends the length of the message by adding 'y' delimiter to the end of the length.

Lets say sender will send 52 bytes data over the socket, then it sends string of "52y" over the socket and reader reads byte by byte until reading character of 'y' and parses number of 52. After that writer sends the data in a loop, since sometimes it cannot send all the data at once. Let's say it sent half of the data, then it sends the other half at second turn of the loop. Reader reads the data in a loop too until it reads 52 bytes of data from the socket.

After each message sending, sender tries to read 1 byte of the data from the socket and reader sends 1 byte of the data through the socket so they know reader read the message end sender sent the message.

4 Main Thread

Main thread basically accepts of the incoming connection requests with accept function and stores file descriptors, those are returned by accept function, in an array. Each index of the array is associated with a thread. Each thread's id indicates the index of the array too. So when a thread is awaken by the main thread, it gets file descriptor of the socket from that array.

5 Threads in the Thread Pool

When a threads is awaken by the main thread it enters in a loop. Then it gets the socket file descriptor from the array and parses the query that are sent by the client. How they communicate part is mentioned on "Communication Over Socket" section. After it receives the query it parses it and according to the type of the query (SELECT OR UPDATE) it enters reader-writer paradigm. It checks number of readers and writers in the process or waiting, if conditions are satisfied then it passes the monitor. Then it do necessary processes on the data set and stores the result of the query as a string and sends it back to the client. If client sends character of 'n' at the end of the loop, thread recognize all the queries are sent so it can leave the loop.

6 Client

Client basically just reads the associated queries from the file and sends them to the server via socket, gets the response and prints it.