Gebze Technical University
Tugay Apaydın
1801042081

CSE344 – System Programming
Final Project - 2021

FINAL PROJECT REPORT

Commands

Build Server and Client: Make
Run Server: ./server -p PORT -o pathToLogFile -l poolSize -d datasetPath
Run Client: ./client -i id -a 127.0.0.1 -p PORT -o pathToQueryFile

Main Algorithm Design of the "Server"

The server starts after a couple of steps. These steps are making the server daemon, making sure that the server is the only instant in the system (by creating a named semaphore), setting signal handler, reading commandline arguments and opening the log file.
After the server started, it firstly connects to the socket (socket, bind and listen steps). After that, it accepts client connections and finds available threads to send them these connections.

When a thread is created, it firstly sets it's status to 'AVAILABLE'. After that is waits for a signal from the server (By locking a mutex) which means that a connection is given to the thread. After the connection signal arrives, thread firstly read client query. After that, it calls the "query_parser"function (the details are given in the database section). After that, thread writes the size of the message, the number of records and returned table output.

Main Algorithm Design of the "Client"

The client starts right after it connects to the server by reading the query file. It reads the lines in the query file one by one and parses the first token of the lines. If the first token (which represents the client id) is equal to the id of the client, it sends the query to the server.

After client sends the query to the server, it waits for the server response. The first 20 bytes represents the message size and then the next 20 bytes represents the total number of records. After reading these stats, the client reads the response of the server block by block in order to not to cross socket buffer limit.

Main Algorithm Design of the "Database"

The CSV table is built by using 3D char array (Column*Row*Field). Also database includes extra fields in order to restrict access (Reader-Writer paradigm implemented as explained in our lecture slides) such as active_reader, active_writer etc.

The main function of the database is 'query_parser'. It takes the query as an argument and parses according to keywords (SELECT, DISTINCT, UPDATE, WHERE, column names and values etc.). The SELECT and UPDATE query functions basically checks each required columns and rows and then returns the output. The difference here the SELECT DISTINCT function. Dynamically it firsts marks the unwanted rows and then returns the table accordingly.

The implementation of the restrictions to the table (for readers and writers), is implemented as explained in lecture slide 10. There is no big difference in this part.

Requirements

Requirements that are done:
- In case of an interrupt signal, all threads can exit gracefully by freeing all resources and closing file descriptors.
- Synchronization is done by using monitors.
- The server is a daemon and double instantiating is prevented.
- All commands (SELECT, SELECT DISTINCT, UPDATE) are implemented.
- There should not be any memory leaks according to tests.
- The input file is not modified.
- According to test results, all calculation results are correct.

All requirements are done and there are not any failure conditions expected in the execution of the program.

Test Notes

A test CSV file and a query file will be added to the main file.
In my implementation, whitespaces are not considered and they can be dangerous for parsing. For example; for an UPDATE query, instead of this command: Column='Value', if you give this command: Column = 'Value', then server thread cannot parse the input. Whitespaces should be removed.