CSE344 - System Programming - Final Report

Fatih Kaan Salgır - 171044009

Design Explanation

Synchronization

Main Thread & Worker Threads

Main thread creates the worker threads, accepts the incoming connection from clients and put it into `client_queue`. As the worker threads are created, they must wait for `client_queue` to be filled, if the queue is empty. Producer & consumer paradigm is used to solve the problem. However the main thread will not wait for the queue to be empty. Instead, it is going to ignore the requset. Because main thread shouldn't be blocked, it must wait for new connections to arrive.

```
// Producer - main thread
while (true) {
  int client_fd =
      accept(server_fd,
      (struct sockaddr *)&client_addr,
      &addrlen);
  pthread_mutex_lock(&pool.lock);
  offer(&client_queue, client_fd);
  pthread_cond_broadcast(&pool.cond);
  pthread_mutex_unlock(&pool.lock);
}
```

```
// Consumer - worker threads
while(true) {
  pthread_mutex_lock(&pool.lock);
  no_busy--;
  while (client_queue.size == 0)
    pthread_cond_wait(&pool.cond, &pool.lock);
  int client_fd = poll(&client_queue);
  no_busy++;
  pthread_mutex_unlock(&pool.lock);
  // fulfill request and respond
}
```

Readers & Writers

Readers & writers paradigm is used to synchronize database access on `SELECT` or `UPDATE`. After the client connection dispatches to thread, client sends the requested query to thread. Thread parses the query, and according to type of the query (`SELECT` or `UPDATE`), it will behave as either reader or writer. Condition variables (`ok_to_read`, `ok_to_read`), state variables (`no_active_readers`, `no_active_writers`, `no_waiting_readers`, `no_waiting_writers`) and the mutex (`rw_lock`) created as global variables.

Dataset

When the server starts dataset loads into memory. Since add and remove operations are not supported, the size will be static. Therefre array structure is prefered to store dataset.

In `csv_reader.h`;

```
typedef char **record;

struct sql_table {
  size_t size;
  record header;
  record *records;
  size_t no_cols;
};
```

- Both row and column sizes are static, defined in `csv_read.h`.

Query

Parsing the query

Query string tokenized and read into the `query` struct. During the parsing, different number of tokens can be read, or not. If the query string is not valid, `parse_query()` function prints error message and returns -1. In order make the error checking more readable, `SET_IDX()` macro is used.

```
enum sql_command { UNKNOWN = 0, SELECT, SELECT_DISTINCT, UPDATE };

typedef struct pair {
  char *key;
  char *value;
} pair;

typedef struct query {
  enum sql_command cmd;
  bool distinct;
  char *columns[MAX_F];
  pair column_set[MAX_F];
  pair column_filter[MAX_F];
} query;
```

- Queries with ',' character might not work properly, since the delimeter is ',' is used to tokenize query string.
- String compare operations are sensitive to '"' character.

Running the query

`run_query()` function returns the resulting records as an array of indices and the size of the array. Worker thread iterates over the array, reads a row into a buffer (`sprint_row()`), sends the buffer to client, repeats for every record in the array.

Socket Communication

- Backlog arguement of `listen()` call, set to `SOMAXCONN`, which is maximum number of limit defined in `<sys/socket.h>`.
- In case client closes the socket, the server gets `SIGPIPE` signal. It is ignored to prevent server from terminating when the signal arrives.

Preventing Data Loss

It may happens that server cannot send the whole buffer at once. `write()` and `read()` functions returns number of bytes sent/read. To make sure that the data is sent successfuly, the worker thread first sends the size of the buffer, then loops over `write()` until all the bytes are sent. Similirlay the client will read the buffer size first and then the respond string.

For this purpose; `send_int()`, `receive_int()`, `send_line()`, `receive_line()` functions are implemented in `socket_io.c`.

Single Instance

`/tmp/cse344-171044009.pid` file is created and tred to obtain file lock. If the program cannot get the lock, that means there is already an instance running. In this case it will print a message and exit.

Handling SIGINT

In case of receiving `SIGINT`, handler function is sets the global variable `got_sigint`. Since `accept()` calls returns on signal arrives, the threads checks the variable on every iteration, and if it is set the loop breaks. Main thread broadcast on conditional variable to let other threds know `SIGINT` arrived. Resources are freed in exit handler.

Running & Test Cases

- `fprintf` buffer is disabled, to be able to load the contents of the file instantly.

Program tested with different queries and tables with the valgrind to check memory leaks. According the output of the valgrind on both client and the server side, "All heap blocks were freed – no leaks are possible".

Server;

```
[2021-06-10 04:35:53] Executing with parameters:
[2021-06-10 04:35:53]         -p 8082
[2021-06-10 04:35:53]         -o ./logfile
[2021-06-10 04:35:53]         -l 3
[2021-06-10 04:35:53]         -d machine.csv
[2021-06-10 04:35:53] Loading dataset...
[2021-06-10 04:35:53] Dataset loaded in 0.000809 seconds with 201 records.
[2021-06-10 04:35:53] A pool of 3 threads has been created
[2021-06-10 04:35:53] Thread #0: waiting for connection
[2021-06-10 04:35:53] Thread #1: waiting for connection
[2021-06-10 04:35:53] Thread #2: waiting for connection
[2021-06-10 04:36:17] A connection has been delegated to thread id #0
[2021-06-10 04:36:17] Thread #0: received query 'SELECT * FROM TABLE;'
[2021-06-10 04:36:17] Thread #0: query completed, 201 records have been returned.
[2021-06-10 04:36:17] Thread #0: received query 'UPDATE TABLE SET Period='-1' WHERE
↪  Series_reference='BDCQ.SEA1AA';'
[2021-06-10 04:36:18] Thread #0: query completed, 38 records have been returned.
[2021-06-10 04:36:18] Thread #0: received query 'SELECT DISTINCT Series_reference,Series_title_5
↪  FROM TABLE;'
[2021-06-10 04:36:18] Thread #0: query completed, 8 records have been returned.
[2021-06-10 04:36:18] Thread #0: waiting for connection
[2021-06-10 04:36:29] A connection has been delegated to thread id #2
[2021-06-10 04:36:29] Thread #2: received query 'UPDATE TABLE SET Period='-1' WHERE
↪  Series_reference='BDCQ.SEA1AA';'
[2021-06-10 04:36:29] Thread #2: query completed, 38 records have been returned.
[2021-06-10 04:36:29] Thread #2: waiting for connection
[2021-06-10 04:36:44] A connection has been delegated to thread id #1
[2021-06-10 04:36:44] Thread #1: received query 'SELECT DISTINCT Series_reference,Series_title_5
↪  FROM TABLE;'
[2021-06-10 04:36:44] Thread #1: query completed, 8 records have been returned.
[2021-06-10 04:36:44] Thread #1: waiting for connection
```

Client-1;

```
[2021-06-10 04:36:17] Client-1 connecting 127.0.0.1:8082
[2021-06-10 04:36:17] Client-1 connected and sending query: 'SELECT * FROM TABLE;'
[2021-06-10 04:36:17] Server's response to Client-1 is 201 records, and arrived in 0.500690
↪  seconds
[2021-06-10 04:36:17] 0 Series_reference | Period | Data_value | Suppressed | STATUS | UNITS |
↪  Magnitude | Subject | Group | Series_title_1 | Series_title_2 | Series_title_3 |
↪  Series_title_4 | Series_title_5 |
[2021-06-10 04:36:17] 1 BDCQ.SEA1AA | 2011.06 | 80078 | (null) | F | Number | 0 | Business Data
↪  Collection - BDC | Industry by employment variable | Filled jobs | "Agriculture Forestry and
↪  Fishing" | Actual | (null) | (null) |
...(truncated)
[2021-06-10 04:36:17] Client-1 connected and sending query: 'UPDATE TABLE SET Period='-1' WHERE
↪  Series_reference='BDCQ.SEA1AA';'
[2021-06-10 04:36:18] Server's response to Client-1 is 38 records, and arrived in 0.500673
↪  seconds
...
[2021-06-10 04:36:18] Client-1 connected and sending query: 'SELECT DISTINCT
↪  Series_reference,Series_title_5 FROM TABLE;'
[2021-06-10 04:36:18] Server's response to Client-1 is 8 records, and arrived in 0.500875 seconds
...
[2021-06-10 04:36:18] Total of 3 queries were executed, client is terminating.
```

Client-2;

```
2021-06-10 04:36:29] Client-2 connecting 127.0.0.1:8082
[2021-06-10 04:36:29] Client-2 connected and sending query: 'UPDATE TABLE SET Period='-1' WHERE
↪  Series_reference='BDCQ.SEA1AA';'
[2021-06-10 04:36:29] Server's response to Client-2 is 38 records, and arrived in 0.500790
↪  seconds
[2021-06-10 04:36:29] 0 Series_reference | Period | Data_value | Suppressed | STATUS | UNITS |
↪  Magnitude | Subject | Group | Series_title_1 | Series_title_2 | Series_title_3 |
↪  Series_title_4 | Series_title_5 |
[2021-06-10 04:36:29] 1 BDCQ.SEA1AA | -1 | 80078 | (null) | F | Number | 0 | Business Data
↪  Collection - BDC | Industry by employment variable | Filled jobs | "Agriculture Forestry and
↪  Fishing" | Actual | (null) | (null) |
...
[2021-06-10 04:36:29] Total of 1 queries were executed, client is terminating.
```

Client-4;

```
[2021-06-10 04:36:44] Client-4 connecting 127.0.0.1:8082
[2021-06-10 04:36:44] Client-4 connected and sending query: 'SELECT DISTINCT
↪  Series_reference,Series_title_5 FROM TABLE;'
[2021-06-10 04:36:44] Server's response to Client-4 is 8 records, and arrived in 0.500938 seconds
[2021-06-10 04:36:44] 0 Series_reference | Series_title_5 |
[2021-06-10 04:36:44] 1 BDCQ.SEA1AA | (null) |
[2021-06-10 04:36:44] 2 BDCQ.SEA1AA | ney |
[2021-06-10 04:36:44] 3 (null) | (null) |
[2021-06-10 04:36:44] 4 BDCQ.SEA1BA | (null) |
[2021-06-10 04:36:44] 5 BDCQ.SEA1CA | (null) |
[2021-06-10 04:36:44] 6 BDCQ.SEA1DA | (null) |
[2021-06-10 04:36:44] 7 BDCQ.SEA1EA | (null) |
[2021-06-10 04:36:44] 8 BDCQ.SEA1FA | (null) |
[2021-06-10 04:36:44] Total of 1 queries were executed, client is terminating.
```