NetWork Database

Yutong Ji

Dec 2022

Contents

1 Introduction

The goal of this project is to develop a networking and multithreaded program that handles small weight modern database operations. There are two programs. The first one is **host** - a networked database host (creator) that receive requests from its clients, where clients are able to store, get, put, modify, delete string-based tuples (the tuple can be any dimensions), and some smarter search - *basically, cover a fair bit of core operations on SQL* (later on in the advanced version). The other one is **client** - a networked client (user) that can query on the host with proper securities such as authentication and the max number of connections. Later on, if have time, there will be multiple-processing included to handle multiple hosts (more than one database) and more advanced functions (see Advance section).

2 Implementation Detail

2.1 host

The multithreaded host uses a command-line interface to open an available port unless keyboard interrupt (Ctrl-C) has been pressed, it will continue to open the port and receive the requests from its clients.
Requests:

- The **store** request allows a client to store a tuple (could have NULL values in some field, except key field) contains the required key. If the key value is already existed in the database, then its other value will be replaced.
- The **retrieve** request allows a client to get a tuple from the database by providing its key value, if the tuple does not existed, error message will be printed in the terminal.
- The **delete** request allows a client to delete a tuple from the database by providing its key value, if the tuple does not existed, error message will be printed in the terminal.

2.1.1 Command Line Detail

The host program only accept the command line arguments:

```
./host authfile connections type [port number]
```

Note: [port number] means it is an optional argument, so if the port number if not provided, then host should attempt to open an port that is currently available.

If the program receives an invalid command line, then it will print following message to **stderr** and exit with code **1**:
Case1:
Invalid comment line usage

```
Error: InvalidUsageError
Correct Usage: ./host authfile connections type [port number]
```

Potential invalid command line:

- authfile or connections are not provided or provided in the wrong-order
- connections is a negative number
- port number is not an integer, if provided, or it is not in range $[1024, 65535]$

- too many arguments
- type is not one of the followings: private, public.

Note: if the type is public, then the provided authfile is ignored.

Case2:
If the authfile is an invalid file to open for reading, or it is an empty file

```
Error: InvalidFileError, fail to obtain the authorisation information
```

Case3:
If the specified port number, if provided, is occupied at the moment

```
Error: InvalidSocketError, fail to open and listen to socket {port number}
```

2.1.2 After Command Line

Before a port is open for listening, in the terminal, a series user input should be conducted to initialised a database first:

```
Database name:
Number of column:
Key field column name:
Column name:
Column name:
...
```

If those information are provided correctly, a successful message should be printed in **stdout**:

```
The database is listening on port: [port number]
```

Otherwise, an corresponding error message should be printed in **stderr** and the program exits with code **2**. Note: no input field can be empty, no column name can be repeated, and "Number of column name" has to be an integer that ≥ 1

After that, the host should open the port and starting to listen and response to any incoming request from clients.

2.2 client

The client uses a command line interface as well to invoke the program to send requests (mentioned in host section) to the host.

2.2.1 Command Line Detail

The client program only accept following arguments:

```
./client portNumber operation
```

Where, portNumber states which localhost port of the host is listening on, operation has to be one of the followings - store, retrieve and delete.
If the program receives an invalid command line, then it will print following message to **stderr** and exit with code **1**:

Case1:
Command line usage error

```
Error: InvalidUsageError
Correct Usage: ./client portNumber operation
```

Potential invalid command line:

- too few/many arguments
- operation is not in one of the followings - store, retrieve and delete.

Case 2:
If the provided port is wrong:

```
Error: InvalidPortNumberError, fail to connect to port {portNumber}
```

3 Communication Guidelines

The communication between host and client follows some specific format. The connections between them is kept alive before the request finished. Moreover, the action between host and client is synchronous - one client can only access the host at any time to enhance the concurrency and accuracy of the operation between them.

3.1 client to host

3.1.1 store request

When the operation in the client command line is "store", if the connected host's type is private, then the client will be asked to input authfile code first.

```
Authorization Code:
```

If the code is not matched, then the program exits and following error message will be printed in **stderr**

```
500
The passcode is incorrect! Access is denied!
```

After the validation succeed, a series of prompts will be conducted to obtain the column value, if the key field already existed, then the rest column value will be replaced, and a warning message will be printed in **stdout** indicates that a tuple has been updated.
Series of prompts:

```
{Key field column name} : [value]
{Column name}: [value]
{Column name}: [value]
...
```

3.1.2 retrieve request

Similarly with the store request, when the operation in the client command line is "retrieve", if the connected host's type is private, then the client will be asked to input authfile code first (see above).
After that, one prompt will be waiting for user to input:

```
Key value:
```

Note: at this early stage, the user can only obtain the tuple with the key value.

If the key is not existed in the database, then the program exits and following error message will be printed in **stderr**

```
404: Not Found, the entry is not found
```

3.1.3 delete request

Similarly with the store request, when the operation in the client command line is "delete", if the connected host's type is private, then the client will be asked to input authfile code first (see above).

After that, one prompt will be waiting for user to input:

```
Enter the value for [column name] column:
```

Note: at this early stage, the user can only delete one tuple at a time with the key value.

If the key is not existed in the database, then the program exits and following error message will be printed in **stderr**

```
404: Not Found, the entry to be deleted is not found
```

3.2 host to client

Once host received the request from its client, the host should attempt to operate and send back some information to the client to indicate if the operation is successful or not.
Sent back message format:

```
Status
Returned tuple
```

Where, status is one of the following {200, 404, 500} 200 means OK, 404 means key not found, and 500 means unexpected error (in this project, it means the passcode is incorrect specifically).

When client reviewed those sent back message, it should print the associated message in the terminal (see above or example below).

4 Demo

In the demo section, we will create a database named "Student". It stores student number as the key field, student name, student age, and student's degree. And the usage of this project will be demonstrate.

Suppose there is a file named "code.txt" and the first line in that file is:

```
password123
```

4.1 Create the host

4.1.1 public

If the optional argument port number is not specific, the system will fetch a random one that is currently available. And if the type of this host is public, then the system will not check the file provided.
Valid:

```
yutongji@Yutongs-Air Network Database % ./host foo 3 public
Database Name: Student
Number of columns: 4
Key field column name: Student Id
Column name: Name
Column name: Age
Column name: Degree
The database is listening on port: 62458
```

On the other hand, we could also provided a port number, then the system will intend to listen on it instead.

```
yutongji@Yutongs-Air Network Database % ./host foo 3 public 2002
Database Name: Student
Number of columns: 4
Key field column name: Student Id
Column name: Name
Column name: Age
Column name: Degree
The database is listening on port: 2002
```

Invalid: please see above documentation part for detail.

4.1.2 private

If the host type is private, the provided file will be checked. Valid:

```
yutongji@Yutongs-Air Network Database % ./host code.txt 3 private 2002
Database Name: Student
Number of columns: 4
Key field column name: Student Id
Column name: Name
Column name: Age
Column name: Degree
The database is listening on port: 2002
```

Invalid:

```
yutongji@Yutongs-Air Network Database % ./host foo 3 private 2002
Error: InvalidFileError, fail to open the file
```

4.2 Connect to the host by using client

From now on, we will assume that our host is:
yutongji@Yutongs-Air Network Database % ./host code.txt 3 public 2002
Or
yutongji@Yutongs-Air Network Database % ./host code.txt 3 private 2002

4.3 Retrieve operation

Initially the host database has no student.

4.3.1 public

```
yutongji@Yutongs-Air Network Database % ./client 2002 GET
Successfully connecting to the host!
Enter the searching key for [ Student Id ] column: 1
404: Not Found, the entry is not found
```

4.3.2 private

The only speciality about private host type is just inputing password before any operation happened.

```
yutongji@Yutongs-Air Network Database % ./client 2002 GET
Successfully connecting to the host!
You are accessing private database.
Please enter the passcode: password123
Enter the searching key for [ Student Id ] column: 1
404: Not Found, the entry is not found
```

If the password is wrong, client will be disconnected with the host

```
yutongji@Yutongs-Air Network Database % ./client 2002 GET
Successfully connecting to the host!
You are accessing private database.
Please enter the passcode: wrong password
500
The passcode is incorrect! Access is denied!
```

4.4 Store operation

Storing student Tim who is a 20 year-old student with student id 1 and studying Computer Science.

4.4.1 public

```
yutongji@Yutongs-Air Network Database % ./client 2002 PUT
Successfully connecting to the host!
Enter the value for [ Student Id ] column: 1
Enter the value for [ Name ] column: Tim
Enter the value for [ Age ] column: 20
Enter the value for [ Degree ] column: Computer Science
```

Now we have some data in the host database, so we can try to retrieve it.

```
yutongji@Yutongs-Air Network Database % ./client 2002 GET
Successfully connecting to the host!
Enter the searching key for [ Student Id ] column: 1
200: OK, the entry is found! They are:
 Student Id: 1
 Name: Tim
 Age: 20
 Degree: Computer Science
```

4.4.2 private

Similarly as above

4.5 Delete operation

We are going to delete the student Tim.

4.5.1 public

```
yutongji@Yutongs-Air Network Database % ./client 2002 DELETE
Successfully connecting to the host!
Enter the searching key for [ Student Id ] column: 1
200: OK, the entry has been deleted
```

Intuitively, we cannot retrieve the student Tim after deleting it.

```
yutongji@Yutongs-Air Network Database % ./client 2002 GET
Successfully connecting to the host!
Enter the searching key for [ Student Id ] column: 1
404: Not Found, the entry is not found
```

4.5.2 private

Similarly as above

5 Hidden feature - Host database report

The host will prevent most of the signal interrupt except force quit (ctrl-C). When the SIGHUP (signal number one) has been sent to the host, it will print a brief report about the database. After a few operations:

```
yutongji@Yutongs-Air Network Database % kill -1 16423
```

On host terminal:

```
Database Name: Student
Number of columns: 4
Key field column name: Student Id
Column name: Name
Column name: Age
Column name: Degree
The database is listening on port: 2002
Database Name: Student
Total number of connecting clients:0
Total number of clients that completed their operations:2
Total number of failing access private database:0
Total number of GET operations:1
Total number of PUT operations:1
Total number of DELETE operations:0
```

6 Caveat

- since using thread to handle incoming clients, the host will continue to listening to the open port, unless host has been force quit so that there will be some memory allocated cannot be freed during the host is running. Hope the system would free those allocated memory when the host terminate.
- since how client and host communicate is going through a few read/write, and there is 100 characters buffer limits, it may not be able to store some overly large text. The recommend size (sum all characters in each column per entry) is around 80 characters - a lite database.

7 What to be expect further

1. smarter search: being able to search multiple entries in one transaction such as range search, first 10 row after sort and so on
2. stronger restriction: each column would be limited to a certain type (integer, string, boolean) and size
3. memorisation: being able to saving as a file when quit so that the data would not be lost, and also being able to support load operation for example loading a saved file
4. more than one database: fork the shell and being able to handle multiple database at a time