Lab Project 1

(Due date: 4/3 Wednesday, 11:59pm.)

In this lab project you are going to leverage a KV store (specifically LevelDB) to build a scaled-down database system, named TinyTable. Different from implementation of a conventional DB system, TinyDB stores and accesses its data in a table as key-value pairs, each corresponding to a cell in the table. For a specific cell, the key is a combination of the cell's row name and column name (both are character strings), and the value is also a character string that is stored in the cell. As a DB system, TinyTable allows a user to type below commands, one at a time, to define a table, write (set) individual rows, read (get) one or multiple rows, access individual cells, and exit.

1. TinyTable’s Commands

1: table column-name-1 [column-name-2 ..., column-name-n]

Note: This command defines the column names, each a character string, of the table currently maintained by TinyDB. By default, TinyTable maintains only one table. This command must be executed before a table can receive its data for storage. This is especially the case for commands associated with entire rows (without explicitly specifying specific column names) such as ‘setrow’, ‘getrow’, and ‘delrow’.

2: setrow row-name column-1-value [column-2-value ..., column-n-value]

Note: This command writes (sets) values of a specific row (‘row-name’) into the table. The list of column values must match, in terms of the column count and order, the list of column names specified in the 'table' command.

3: getrow row-name-1 [row-name-2 ..., row-name-k]

Note: This command reads (gets) values of each of the listed rows (‘row-name-i’) and display the values line by line.

4: delrow row-name-1 [row-name-2 ..., row-name-k]

Note: This command removes (deletes) values of each of the listed rows (‘row-name-i’).

5: setelem row-name column-name column-value
Note: This command writes (sets) value ('column-value') at a specific row ('row-name') and a specific column ('column-name').

6: getelem row-name column-name

Note: This command reads (gets) value ('column-value') at a specific row ('row-name') and a specific column ('column-name'), and display it.

7: delelem row-name column-name

Note: This command deletes value ('column-value') at a specific row ('row-name') and a specific column ('column-name').

8: exit

Note: This command closes the program.

2. Setup of the experiment platform

First, login to the development server.

The development server runs a Linux operating system (CentOS 7.2).

You need to use an SSH client to login to the development server.

```
hostname: 129.107.35.245
port: 8080
username: your NetID (e.g., abc1234)
password: your NetID
```

[If you connect to the network via UTA’s wireless service, you need to first log into UTA's vpn. Please read "student installation" AND "Employee configuration" for vpn usage. The Employee configuration is also applicable for the student’s setup https://www.uta.edu/oit/cs/software/vpn/mac.php]

Example in Linux/Mac:
```
$ ssh abc1234@129.107.35.245 -p 8080
```

It will prompt for password.
Make sure to use “passwd” command to change your password immediately after your first login.
If you’re in Windows, try putty (https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe). The development server provides (at least) two text editors --- vim and nano. If you’re not sure, use nano. For example:

$ nano 1.txt

Once you have access to the server, extract the lab1 package and build TinyTable:

$ tar -xf lab1.tar.gz
$ cd lab1
$ make -j32
$ ./tinytable

Welcome to TinyTable Console!
Commands:
  Define THE table scheme  $ table <column-1> <column-2> ...
  Print current scheme     $ table
  Get row(s) by row-key(s) $ getrow <row-key-1> <row-key-2> ...
  Set one row              $ setrow <row-key> <value-1> <value-2> ...
  Delete row(s)            $ delrow <row-key-1> <row-key-2> ...
  Get an element           $ getelem <row-key> <column-key>
  Set an element           $ setelem <row-key> <column-key> <value>
  Delete an element        $ delelem <row-key> <column-key>
  Exit                     $ exit

=======
tinytable >

3. Your assignments

1. Carefully read 'tinytable.cc', which is the source code for implementing the set of TinyTable’s commands, and a main() function for providing a console receiving and interpreting commands. ‘sample-input.txt’ contains a list of sample commands for you to familiarize yourself with the system. To batch-run all commands in the file, you can redirect the program’s input to the file: “./tinytable < sample-input.txt”.

In ‘tinytable.cc’, functions prefixed with "cmd_" are for implementing corresponding TinyTable commands. Functions prefixed with "lldb_" are wrapper functions for corresponding LevelDB operations (GET/PUT/DELETE).

2. There are four functions (cmd_getrow(), cmd_delrow(), cmd_setelem() and cmd_delelem()), that are used to implement the corresponding commands, are not completed. Please refer to the implementations of similar functions to complete them.

3. Currently each time you start the program you need to run the 'table' command to supply the list of column names, which are stored in the ‘vector<string> columns’ in the 'struct TinyTable' but are not saved on the storage.
struct TinyTable {
    leveldb::DB * lldb; // low-level LevelDB
    vector<string> columns; // column names
};

To run 'tinytable' without having to use the 'table' command again (suppose you don’t need to modify the list of columns), please implement two functions called in the main() function before the program exit (save_table_scheme()) and at the beginning of the program (load_table_scheme()). Specifically, save_table_scheme() stores the data in the ‘vector<string> columns’ into the LevelDB store, and load_table_scheme() loads the saved column names from the store into ‘vector<string> columns’ (if they have been saved). Be careful about the corner cases. save_table_scheme() should never destroy user’s data stored in tinytable.

4. Submit your work

Make sure you are in lab1’s directory. type “submit1” in the shell and your “tinytable.cc” will be submitted. Note that only a copy of tinytable.cc will be used for grading. Don’t use extra files in your coding.

$ submit1

Each submission will be marked with a timestamp. You can have multiple submissions before the deadline. The last submission will be graded. However, the submissions beyond the deadline will be discarded. DO NOT submit your work via email.