Version 1: basic function built up in the table class like add_record, delete_record , print_table....,the record class was pretty much the same, I was also deciding to use a file class to manage the load and save function, built a prototype of loading function which require an “end” string in the end of the file

Version 3: decide to put the write file and load file function inside the Tables class, becuase I was thinking that each table is an object and I should be able to load file to create a new table object and each table can write into a file to save its own data. If I put load and write function inside file class, these functions won’t be able to get access to the data of the tables which are private inside the table class. Therefore, I kept these functions inside the table class instead so they can access the data of individual table object

Version 4: experimenting on extending the record class for the Tables class to use all the function in the Records class. (found out later on this is unnecessary)

Version 5: Implemented the database class, like add table, add table by file, delete file, basically most of the function in the table class. I was planning the use only need to create the database class object, and all the function inside the database class will create the require table object which will create the record object. In the table class, I tried to implement Scanner to get user input when the functions were called.

Version 6: Implementing the idea of having a parser in the database class so user can use code to control the database. The table class was still using Scanner to get user input, it needed to be changed to work with the parser in the Databases class

Version 7: The table class updated the way to get input for the functions, the functions are taking in a String[] array now, as array has to has a finite size so most of my loading process was done by load in the text into a linkedlist and then turn the linked list into a string array, and the parser to understand the code base on the positions of the words in the array. Therefore , spaces are very important to separate the words. Database class now separate the parser as two part, one is to interpret the command and do the action which is command(), and the parser() function to use recursion to create a loop which will not end until the user type in EXIT_DATABASE. Also , I implement the file_parser() which allow user to load in a txt file of code to do all the actions to the database

Version 8: Table class introduced the type check and part of the foreign key check. Database class introduced the other half of the foreign key check. Also introduced another linked list to store the row name data type in the tables class

Final version: final touch of refinement on the foreign key check, now it is valid when multiple column has foreign key. Also make sure all the commands has try and catch. Add in the unit testing.

Design of the database

4 class in total, Records class, Tables class, Databases class and main class. According to some of my online research[1] , I found out that LinkedList is faster in add and remove, but slower in get. As a database involve a lot of add and remove data so I pick linked list as my foundation of the database.

Records class:

- Main purpose is to hold a linked list which hold all the data
- Each record which each row of the table is also will be one object of the Records class
- Only has basic function like add remove data from the linked list

Tables class:

- This holds 2 linked list
- One if to hold multiple Records objects together to create a table, the other one is to hole the field name and the data type for each column. The table very first column is also the field name but without the data type
- When the user wants to add an table, the first thing is to add the field name and the column datatype, therefore the very first line of the data file are the field name and data type
- This class has all the functions to add and delete and alter the records inside the table
- This class also can insert key to the records, the keys are inside the first column of the record only, and the key will always follow the records, not the table
- This class do the data type check , every time a table is load in from file, or user add in a new column or record, the type check will perform
- Also, part of the foreign key existence checking, because the table linked list is private and the Database class require the function in this class to get the information of the table, like check does the table has AUTO_GEN_KEY or not, and copying all the keys of the table and return it back to the Databases class

Databases class:

- This class has two linked list the database linked list takes is taking multiple Tables objects to create a database, the other one table_name records the name of each of the tables, as Tables will only be add to the end of the list so the table_name list and the Database list will have the same order, so the order is used to match the table name with the table
- This class has a file_parser which load in the code text file and turn it into String array in order for the command() function to execute all the commands
- The commands() function takes in an array of string and read the words one by one to execute the command codes. Need another parser() function to use recursion to create a loop so the use can keep stay in the database and make changes until they type in EXIT_DATABASE
- The commands() function is consisted of a switch statement , each case require try and catch because I want the error will not break the loop
- I let the data to load into the table first then I will check the foreign key for validation.

Commands for the database:

!!!!Please remember everything follow with a *spacebar*, when naming stuff like first_name , last_name, use _ instead of spacebar otherwise the data won't be taken in !!!!!!!!

When the data type or the foreign key is wrong, database will turn them into NULL

Data type for column:

INT ---> e.g. age (INT)

FLOAT ---> e.g. Quantity (FLOAT)

ENUM ---> e.g. Country (ENUM jp uk usa fr)

STRING ---> e.g. first_name (STRING)

STRING can be inputted anything. All other three, if the database detects the wrong data type is inputted, it will automatically set that cell as NULL.

AUTO_GEN_KEY

- This provide the automatically generated primary key for the table, it is a number. *If you add auto gen key, and all more record afterward, you don't have to type in the number for the key in the record, just type in the content of the record and the key will be automatically add to the record inside the table*

ADD_RECORD_TO_TABLE data data data .....

- (e.g. ADD_RECORD_TO_TABLE kevin ho 22)

ADD_TABLE_BY_FILE file_name file_path

- (e.g. ADD_TABLE_BY_FILE Test2 /home/kevin/Documents/kevin_test.txt) load in the datafile as a table in the database

ADD_TABLE table_name field_name (data_type) field_name (data_type) .....

- (e.g. ADD_TABLE Test1 first_name (STRING) last_name (STRING) age (INT) gender (ENUM M F) partner (FOREIGN_KEY Test2)) manually create an table, this only create the field name and the data type of each column, table is still empty, records need to be filled one by one

ADD_COLUMN col_destination(int from 0) field_name (string) data data data .... [field_name (data_type)]

- (e.g. ADD_COLUMN 6 Country jp uk hk fr jp usa [Country (STRING)]) ***DONNOT type in the number for auto key, just ignore the auto_key column and fill in the content, the key will be automatically added afterward into the table*** Please don't forget need the field_name at the beginning and add the field name again after the [

DELETE_COLUMN col_name(string)

- (e.g. DELETE_COLUMN first_name)

DELETE_N_COLUMNS start_col_name(string) end_col_name(string)

- (e.g. DELETE_COLUMN first_name last_name)

DELETE_RECORD row_number(int)

- (e.g. DELETE_RECORD 3)

DELETE_N_RECORDS start_row_number(int) end_row_number(int)

- (e.g. DELETE_RECORD 3 6)

DELETE_TABLE table_name

- (e.g. DELETE_TABLE Test1) *you will not be able to delete the very last table left in the database*

EXIT_DATABASE

LOAD_CODE_FROM_FILE file_path

- (e.g.LOAD_CODE_FROM_FILE /home/kevin/Documents/ test.txt) This allow you to load in a file of code to do all the command in one goal, example "file_parser_test.txt" is provide please have a look, please don't mix up with the data txt file, their formats are different, ***also only for the code file, each command need to follow by a ; this is very important***

PRINT_TABLE

- *The row number is not in the records, they are only printed out only for the user to know , they are not embedded inside each records, but the keys are.*

PRINT_TABLE_INFO

- Show the column details of the table like the data type of the column

SHOW_DATABASE

- List all the table in the database right now

SELECT_TABLE table_name

- This let you to select which table in the database you want to use, one you changes, name of your current using table will be shown inside this < > . *Also, once you load in a table or manually add in a table, the selected table will be default the first table in the database, so don't forget to change the table you want to use*

SAVE file_path append_to_file(true/false)

- (e.g. SAVE /home/kevin/Documents/test.txt false) pick false will overwrite the whole file with the new content, pick true will not but the file will not be able to load back into the database

  Don't forget to pick true or false !!!!!!!!

SAVE_AS file_path file_name

- (e.g. SAVE_AS test.txt /home/kevin/Documents/) ***!!!File name goes first !!!!****all other command don't need to separate file name and file path, this allow you to name whatever you want of the new file*
- this will create a new file with the name of the file_name in the file you provide

UPDATE_CELL col_number(int) row_number(int) replacement_string(string)

- (e.g. UPDATE_CELL 1 2 happy)

Example code just copy and pase when you compile the main class :

ADD_TABLE Test1 first_name (STRING) last_name (STRING) age (INT) gender (ENUM M F)

ADD_RECORD_TO_TABLE kevin ho 22 M

ADD_RECORD_TO_TABLE John cena 55 M

ADD_RECORD_TO_TABLE Jessica Jones 25 F

ADD_COLUMN 4 Country jp uk usa [Country (ENUM jp uk usa fr)]

AUTO_GEN_KEY

ADD_RECORD_TO_TABLE Tomb Raider 22 F hk

PRINT_TABLE

SAVE_AS final_writing_test.txt /home/kevin/Documents/

ADD_TABLE_BY_FILE Test2 /home/kevin/Documents/kevin_test.txt

SELECT_TABLE Test2

PRINT_TABLE

PRINT_TABLE_INFO

SHOW_DATABASE

ADD_RECORD_TO_TABLE John Cena Y 45 M 2

PRINT_TABLE

Example to load instruction file :

LOAD_CODE_FROM_FILE /home/kevin/Documents/file_parser_test.txt

Make sure your terminal window is big enough !!!!!!!

1) https://dzone.com/articles/arraylist-vs-linkedlist-vs