Erin Marden

CS 457

Programming Assignment 4

Design Document

My program Implements twelve functions to create a database management system. It has main(), createD(string), delD(), use(string), createT(string), and delT(string), insert(string), split(string, char), select(string), selectPart(), setWhere(), deleteWhere(), tupleJoin() as well as helper .txt files locked.txt, user.txt, function.txt, del.txt, update.txt, temp.txt, tupleSelectFile.txt, set.txt, and where.txt.

main() :

* Creates a map that is able to parse through command line arguments to call the correct function.
* If argc>4, then it creates the function.txt and writes everything after to that file, as long as argv[1] doesn’t equal “set”, “select”, “on” or “where”. It sets argv[3] as the name of the file/folder we want to use/create. If it equals select, then it uses the last argument to pass into the map. If it equals set, it checks to see if the file locked.txt has “locked” or “unlocked” written in it. If it is “locked”, then it creates set.txt and writes argv[2] and argv[4] to it. If it equals on, then it saves argv[2-4] to where.txt and calls tupleJoin(). If it equals where, it checks to see if the file locked.txt has “locked” or “unlocked” written in it. If it is “locked” it checks to see if del.txt and select.txt are open. If del.txt is then it writes argv[2-4] to where.txt and then calls the deleteWhere(). If select.txt is open then it writes argv[2] and argv[4] to where.txt.

If del.txt and select.txt are not open, but tupleSelectFile.txt is then, it saves argv[2-4] to where.txt and calls tupleJoin() If neither are open, it writes argv[2-4] to where.txt and calls selectPart().

* If argc =4, this checks if argv[1] equals “select”, “from”, or “delete”. If any of those are true a .txt helper file is created and argv[2-3] written to it. If it’s not equal then we set argv[3] are the second parameter in the map.
* If argc==3, it checks if argv[1] is equal to either “USE” or “update” or “begin”. If it is “USE”, then it creates user.txt to store what folder we want to use and sets argv[2] as that folder name. If it is “update”, it checks to see if the file locked.txt has “locked” or “unlocked” written in it. If it is “unlocked” then it prints an error message and quits the program. If it is “locked”, it does the same thing as “USE” and creates update.txt and stores argv[2] in it. If it is “begin” then it checks to see if the file locked.txt already exists. If it doesn’t then it creates the file and writes “locked” into it. If it does, it overwrites “locked” to “unlocked”.
* If argc == 2, it checks to see if the user wants to exit or commit. If the user chooses to commit, then it checks to see if the file locked.txt has “locked” or “unlocked” written in it. If it is “locked” then the setWhere() function is called to make the updates, and then lock.txt is removed. If it is “unlocked”, then it aborts because there should be nothing to commit anyway. If the user chooses to exit, then the helper files are deleted and the project exits.

createD(string):

* String being passes is the name of new folder
* Created directory from name
* Deletes all contents of the user.txt

deleteD():

* String being passed is the name of the folder to be deleted
* Using remove() function deletes the folder
* Deletes all contents of the user.txt

use(string):

* It was supposed to be how directories were changed, but I couldn’t make it work properly so now this is empty.
* Kept incase I decide to try again

createT(string):

* Opens function.txt and copies the parameters written to that file to the new file. Parenthesis do not work yet-cannot figure out how to get command line to accept them.
* Opens user.txt and uses what is written there to change to correct folder.
* String passed is the name of the new file

delT(string):

* Opens user.txt and uses what is written there to change to correct folder.
* String being passed is the name of the file to be deleted.
* Using the remove() function to remove the file.

Insert(string):

* Opens user.txt and uses what is written there to change to correct folder.
* Uses the parameter being passed to traverse to the correct file and uses function.txt to append.

Split(string, char):

* This takes in the table values and cuts out the commas from them.
* This also formats them to add the “ | ”.

Select(string):

* Reads the directory from user.txt and changes into that folder.
* Uses the parameter being passed to traverse to the correct file.
* Prints everything from that file.

selectPart():

* Opens where.txt and uses what is written there to set what columns we want to read from.
* Opens user.txt and uses what is written there to change to correct folder.
* Opens from.txt and uses what is written there to change to correct file.
* Finds the appropriate columns and prints them

setWhere():

* **as of right now the setWhere function will only change the first occurrence of the where name.**
* The files set.txt and where.txt save what we want to change where
* Opens user.txt and uses what is written there to change to correct folder.
* Uses the update.txt to read what file we should be working in.
* Finds the lines that should be replaced, and then depending on if we are looking for pid, price or name it replaces that section accordingly.

deleteWhere():

* Uses what is inside where.txt to save what we want to delete.
* Opens user.txt and uses what is written there to change to correct folder.
* Uses del.txt to navigate to the file we should be in.
* Finds the lines that should be deleted, and then deletes that section accordingly

tupleJoin():

* It will read user.txt to figure out what directory to be in
* It will read from from.txt to decide what tables to use.
* It will read from tupleSelectFile.txt to decide what variable to use.
* It will read from where.txt to know what parameters to check for.
* It checks if we doing inner join or left outer join.
* While it reads through file1, it also reads through file2 and if what we are searching for are equal then it prints them out together.

user.txt:

* Holds the folder name the user wants to access

function.txt:

* Holds the list of parameters we want for each table

locked.txt

* Holds whether the program is locked or unlocked

What lines to individually input to make this run:

--terminal 1

./emarden\_pa4 CREATE DATABASE CS457\_PA4;  
./emarden\_pa4 USE CS457\_PA4;  
./emarden\_pa4 CREATE TABLE Flights seatint, statusint;  
./emarden\_pa4 insert into Flights values 22, 0;

./emarden\_pa4 insert into Flights values 23, 1;   
./emarden\_pa4 begin transaction;  
./emarden\_pa4 update flights;

./emarden\_pa4 set status = 1;

./emarden\_pa4 where seat = 22;

--terminal 2

./emarden\_pa4 USE CS457\_PA4;  
./emarden\_pa4 select \* from Flights;  
./emarden\_pa4 begin transaction;  
./emarden\_pa4 update flights;

./emarden\_pa4 set status = 1;

./emarden\_pa4 where seat = 22;  
./emarden\_pa4 commit;  
./emarden\_pa4 select \* from Flights;

--terminal 1

./emarden\_pa4 commit;   
./emarden\_pa4 select \* from Flights;

--terminal 2

select \* from Flights;