Parker Nelms

CS 457 – Project 4

May 10, 2022

Dbmanager.py design document

**Compilation instructions:** In Ubuntu python interpreter: python3 <filepath>/DBmanager.py

**Purpose:** The purpose of this program update is to allow for transaction based table updates and atomicity among tables in the database.

**Organization implementation:** This program uses a boolean that indicates whether or not the program is currently in a transaction. Only once inside of a transaction, the user can make potential changes to a table. The user can then commit those changes, and, as a result, save the changes to the disk. Before the commit, changes were only saved within the program. If the user quits the program or there are no available changes, no commits will take place. While one process has changes queued up for a table, no other process can make changes to that same table. To achieve this functionality, I have added “startTransaction()” and “commit()” as well as updated versions of “update()” and “setData()”.

**Function implementations:** First,“startTransaction()” sets a bool inTransaction to True so that the update/setData function knows if it can store changes for a given table. It then creates dictionary entries that link table names to a bool that say whether or not the current process can edit that table. Initially all tables are set to false. The new update function checks to see if a transaction has been started and then checks if it is a single or multi line command. The setData function, instead of pushing changes to the table itself, it pushes changes to a global list called “transChanges” that stores all changes made during the transaction. It also opens a lock file for a table the first time a change is stored for that table. Data is also copied from the original table to the global list the first time it is updated and all subsequent times, only the list items are referenced. If the process attempts to update the table but a lock file already exists, the changes will be discarded. This is probably redundant as the process’ ability to alter a table could be checked through the dictionary call which keeps track of table owners. Through these two methods, however, the tables within the program achieve atomicity. The select function is the same and will only pull from the unaltered version of the given table. There was also a small change to allow for in-line comments.