Project 1: Metadata Management

Name: project1.py

Author: Mackenzie Zappe
Date: 9/28/2021

CS 457

I decided to organize my database management system along with the example in the project description. I allowed my databases to be reflected as directories and the tables are files within those directories. I wrote my program in python3.

To parse through the commands, I chose to use the standard input. I did this by including the fileinput library and initially designating a list to store all the contents from the test file. I then went through using different lists to extract the pertinent data relating to commands. Once I did this, I split the data into two parts. One part determined the command word data and the other, if applicable, contained the table attribute info. Then I used a series of if/else statements to determine the actions associated with the command words, CREATE, USE, DROP, SELECT, ALTER, and .EXIT. Then for the CREATE and DROP functions, I built nested if/else statements to determine if the actions were to be applied to a DATABASE or TABLE.

For the CREATE DATABASE function, I employed the os library which allowed me to access the working directory of the program. From there I was able to use os library functions such as os.mkdir to create the subdirectories inside the current working directory to act as my databases. There was a check in place to ensure that a database being created didn't currently exist.

For the CREATE TABLE function, I created an 'append' file and then used the table info list to parse through the table attributes. I did this in a list fashion and wrote the attribute names and types to the according file as it was being created. Similarly, to the database creation, I checked if the current table being created didn't already exist in the current directory.

For both the DROP DATABASE and DROP TABLE commands, my implementation was similar. First I checked if the file or directory path existed and then used the os.remove() function. This function only applies to empty directories. If the file or directory no longer existed at the time of attempted removal, I had a statement print out to the terminal.

For the USE DATABASE command, I first had to return to my default working directory, which I designated by a global variable. Then I created a path that included the new database name. I changed the current directory to that path and printed the appropriate response.

For the SELECT * FROM command, I had to create an additional function called 'find'. The 'find' function used os.walk() to traverse all directories and sub directories in the current working directory in search of the name of the file in question. If that file was found somewhere in the working directory tree, the 'find' function would return the current directory where the file was found. This path would then be used to go to aforementioned directory to open and read the file.

For the ALTER command, I broke the new attribute into a name component and a type of component. I then individually appended the name and type to the table that is being altered. For the .EXIT command, I used 'break' to end the looping statements.