CS-457 Database Management Systems

Project 2

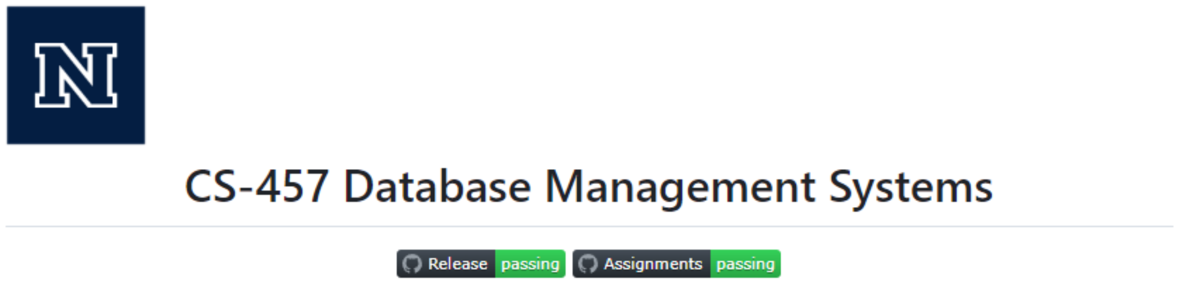
April 3rd, 2023

Install:

https://github.com/Stehfyn/cs457

Contact:

Stephen Foster
stephenfoster@nevada.unr.edu



CS-457 Database Management Systems

Release passing Assignments passing

System Design

The database for this project is designed such that a file is analogous to a database, and that special syntax denotes tables within the database. The following is a visual representation:

```
/databases-+                  db1:
           +-db1              +----------------+
           |                  |[table1]        |
           +-db2              |__rows__ = 1    |
           |                  |__attrs__ = 2   |
           |                  |a4 = int        |
           |                  |a5 = varchar(20)|
           +-db3              |0 = {‘a4’:’2’,  |
           |                  |‘a5’:’hello!’}  |
           |                  |                |
           +-db4              |                |
           |                  |                |
           +-db5              +----------------+
```

As is expressed above, “databases” is a file directory, with db1-db5 being text files with no extension. On the right-hand side of the above visual, we can see tables denoted using angular brackets ([]), with attributes and constraints on the left and right side of the equals (=), respectively. Implementation attributes have been added (__rows__ and __attrs__) to aid database reading and writing. Additionally, one can see how __rows__ is set to the value of 1, denoting the fact that row 0 is populated with the table entry {‘a4’: ‘2’, ‘a5’:‘hello!’}.

System Implementation

The database for this project is written using python3.9+, and has a file structure that is the following:

```
/src-+
     |
     +-main.py
     |
     +-utils.py
     |
     /database-+
               |
               +-database.py
               |
               +-database_impl.py
               |
               +-database_parser.py
               |
               +-embed.py
               |
               +-embedded.py
```

- **main.py -** The database program entrypoint. Depending on how the program was called (interactively, non-interactively, certain command-line arguments, etc.), this file will execute the appropriate functionality found in **database.py**.

- **database.py -** The database program manager. This file contains the DatabaseManager class and related functionality to keep track of which database is in use, as well as the implementations of the batch processor, interpreter, and graphical user interface. Additionally, this file implements the wrapper functions that "connect" the output of the database parser to the appropriate database function.

- **database_impl.py -** The database function implementations. Functions for creating databases, dropping databases, creating tables, dropping tables, etc. exist here. These are the database functions that **database.py** connects with output from **database_parser.py**.

- **database_parser.py -** The database parser that either accepts or rejects raw input as valid database operation language. This parser both tokenizes arguments as well as ensures valid table syntax, then returns the output to **database.py** to connect to the proper database functionality in **database_impl.py**.

- **embed.py** - A helper module that auto-embeds base64 encoded resources such as icon files or typefaces. It overwrites itself and employs **embedded.py** to hold the embedded resources as native python strings. Not strictly necessary for the project but fun to write.

- **test_database.py** - A pytest script that runs the assignment test scripts and compares stdout to the expected stdout. It is hooked up to a Github Action via tox, and has a status badge displayed on the github repository. The currently tested assignment scripts are as1 and as2:

```
@pytest.mark.parametrize("script, stdout",
    [
        ("PA1_test.sql", pytest.lazy_fixture("capture_stdout")),
        ("PA2_test.sql", pytest.lazy_fixture("capture_stdout"))
    ]
)
```

System Interface

Install

For quick install and execution, please clone https://github.com/Stehfyn/cs457. There is a bootstrap script in the repo that creates a virtualenv but at current it only works for Windows. One may use a virtualenv of course, but the manual, no fluff instructions are as follows for Ubuntu:

1. Install python3.9 via:

   ```
   sudo apt-get update && sudo apt-get install python3.9
   ```

2. Install external dependencies via:

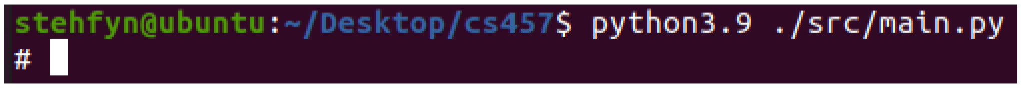
```
python3.9 -m pip install -r requirements.txt
python3.9 -m pip install -r requirements_dev.txt
```

3. Navigate to /src containing main.py and execute:

```
python3.9 ./src/main.py
```

Interpreter

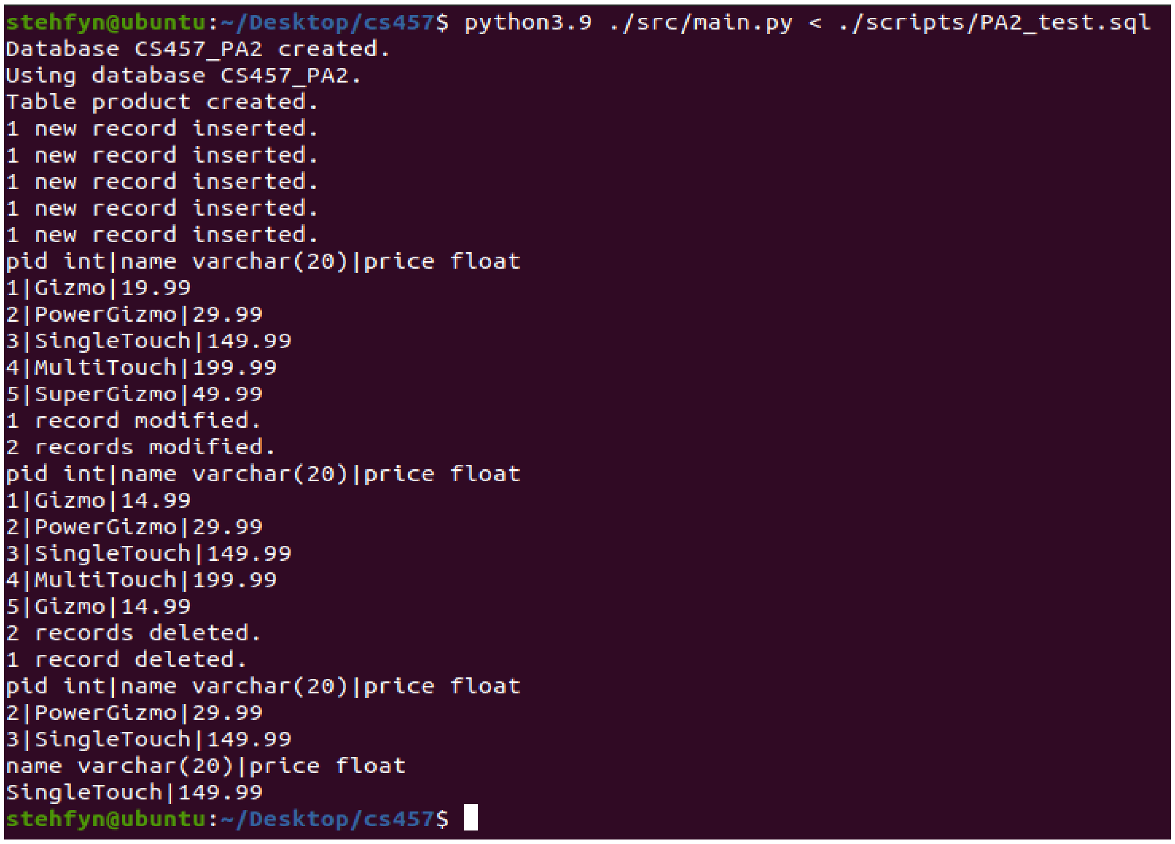
The database project can be executed in interpreter mode, by invoking:



```
stehfyn@ubuntu:~/Desktop/cs457$ python3.9 ./src/main.py
# 
```

Batch Processor

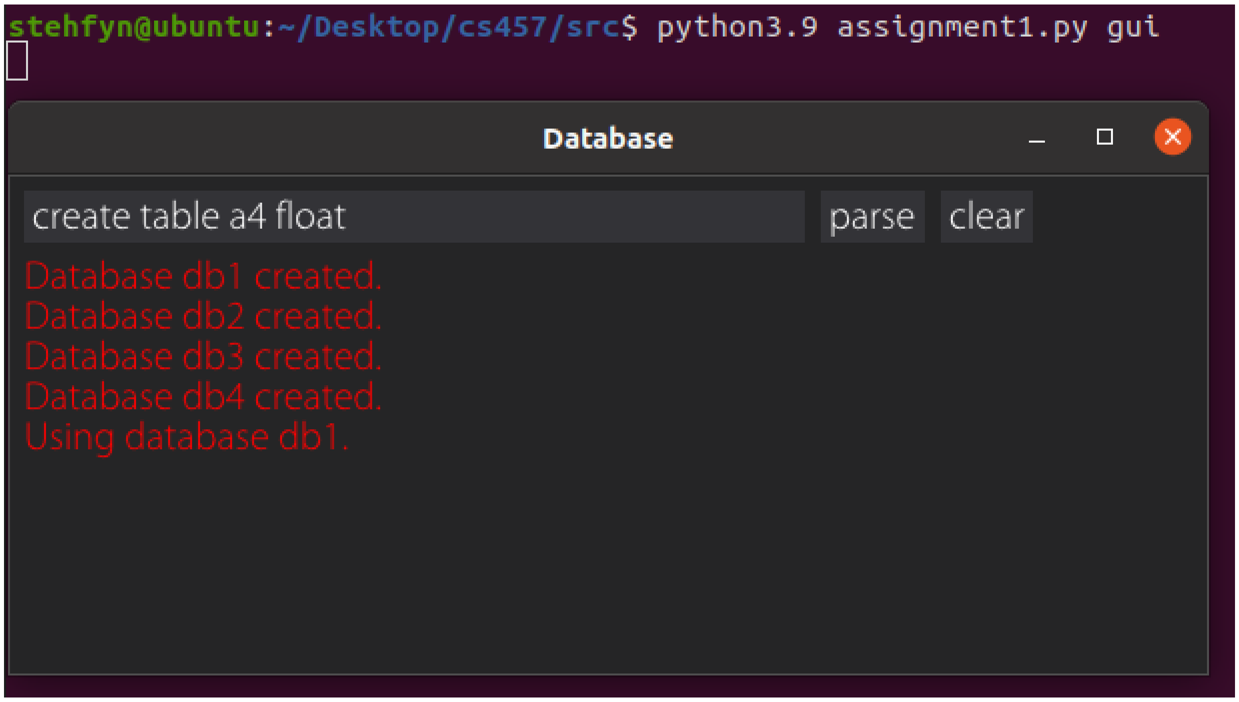
The database project batch processor at present does not only expect filepaths as command-line arguments, and **now does** read in information from stdin. This behavior is accessed by invoking:



```
stehfyn@ubuntu:~/Desktop/cs457$ python3.9 ./src/main.py < ./scripts/PA2_test.sql
Database CS457_PA2 created.
Using database CS457_PA2.
Table product created.
1 new record inserted.
1 new record inserted.
1 new record inserted.
1 new record inserted.
1 new record inserted.
pid int|name varchar(20)|price float
1|Gizmo|19.99
2|PowerGizmo|29.99
3|SingleTouch|149.99
4|MultiTouch|199.99
5|SuperGizmo|49.99
1 record modified.
2 records modified.
pid int|name varchar(20)|price float
1|Gizmo|14.99
2|PowerGizmo|29.99
3|SingleTouch|149.99
4|MultiTouch|199.99
5|Gizmo|14.99
2 records deleted.
1 record deleted.
pid int|name varchar(20)|price float
2|PowerGizmo|29.99
3|SingleTouch|149.99
name varchar(20)|price float
SingleTouch|149.99
stehfyn@ubuntu:~/Desktop/cs457$ 
```

Graphical User Interface

The GUI can be accessed by invoking:



```
stehfyn@ubuntu:~/Desktop/cs457/src$ python3.9 assignment1.py gui
```

```
Database db1 created.
Database db2 created.
Database db3 created.
Database db4 created.
Using database db1.
```

Notable Dependencies

The project relies solely on standard python3.9+ libraries to accomplish the required functionality, and two external libraries to accomplish the graphical user interface. The notable libraries are:

- **argparse -** The tokenizer for the database language, aids in implementing the grammar.
- **configparser -** The file reader/writer. This project leverages its simplicity in reading and writing “sections” of a file through a python dictionary-like programming interface, and is wrapped/extended by **database_impl.py** to implement database reading and writing.
- **dearpygui (external) -** An external graphical user interface library for python, is used by **database.py** to implement a windowed interface to the database functionality.
- **psutil (external) -** An external library to determine the parent process of **assignment1.py**, thus aiding in determining the run mode of the database (interactive, non-interactive, etc.).