DS5010 Course Project Report
Author: Jinhao Lu, Tianbao Piao, Anish Rao
Github Link: https://github.com/JinhaoLu-At-NEU/DS5010_course_project

Summary:

The Handy Datatable package aims to provide a simple and consistent interface for working with data tables in Python. The package includes three modules, namely handy_data_table, handy_graphviz, and handy_sqlite. The handy_data_table module provides functions for creating, modifying, and filtering data tables. The handy_graphviz module provides a simple interface for creating directed graphs to show the relationships between different tables, and finally the handy_sqlite module provides an interface for interacting with SQLite databases. Particularly, this module allows the user to manipulate data within the table by converting simple python commands into sql statements. It can also export the data table as a database(.db) for further manipulation. This package can be useful in various applications such as data manipulation, data visualization, and data storage and retrieval.

Design:

The **'Handy_data_table'** class is the parent class and there are several default parameters that need to be defined when the user initiates a new object. The '__init__' method initializes the table with the parameter table name, list of entities, primary key, and foreign key. The 'add' method adds an entity to the table by specifying its name and data type. And the 'drop' method removes an entity from the table. The 'table_status' method displays the table's name, entities, and their respective data types in a *create table statement*-like format.

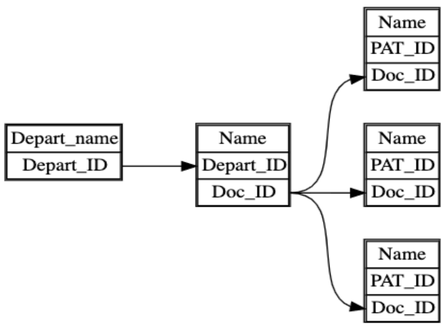
The 'add' method allows users to append new entities to the object. The entity's name should be unique in the entity list, otherwise an error will be raised if input duplicates the entity name. Similar to the 'add' method, the 'drop' method allows users to drop entities from the entity list. If the inputted entity name does not exist in the current object's entity list, there will be a message indicating inputted entity name not found in the entity list.

All of the methods above offer a data table structure object, 'Handy_graphviz' and 'Handy_sqlite are the derived class of the 'Handy_data_table'. The objects initiated under 'Handy_sqlite' and 'Handy_graphviz' inherit the attributes and methods from 'Handy_data_table'.

Handy_graphviz module:

- The 'graphviz_convert_export' method outputs a text file containing dot script, which can be used by the graphviz library for generating visualization of the current viz_data_table object.
- The 'relation_mapping' method adds a mapping relationship between two tables based on the specified entities, the relationships stored in the variable 'mapping_relation' as tuple.
- The 'relation_mapping' method will convert all the tuple values as a directed edge on the visualization.
- The 'clean_relations' method clears all mapping relationships and the 'drop_relations' method removes a specific mapping relationship.

- The graph below is the output of the demo code. The demo code for using ‘handy_graphviz’ module can be found on github repository ‘handy_graphviz_demo.py’*



All the relationships are stored in a global variable ‘mapping_relation’ as a list under the handy_graphviz.py.

Handy_sqlite module:

The ‘create_table’ and ‘insert_table’ methods output a text file containing SQL script, which is the script that can be used to apply DML and DCL operations to the SQLite db file.

The ‘check_value_column’ method applies a validation on the column number between the table and data file. If the number of entities and number of columns doesn’t match, error will be raised.

Usage

Handy_graphviz module

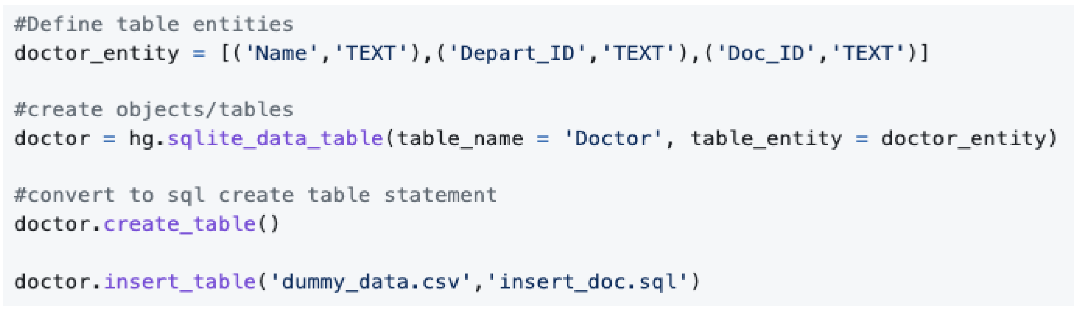
```
#create objects/tables
department = hg.viz_data_table(table_name = 'Department', table_entity = department_entity)
doctor = hg.viz_data_table(table_name = 'Doctor', table_entity = doctor_entity)
patient1 = hg.viz_data_table(table_name = 'PAT1', table_entity = patient1_entity)
```

The user can create tables without SQL but only need to initiate objects with a straight forward string.

```
#Convert and output the table structure into dot scripts
department.graphviz_convert_export('department.txt')
doctor.graphviz_convert_export('doctor.txt')
patient1.graphviz_convert_export('patient1.txt')
```

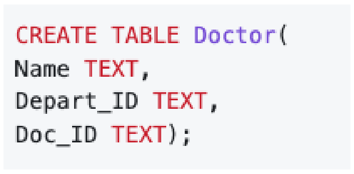
With the graphviz_convert_export method, the content of the object will be converted to the dot script into a text file. Then the user can read the text file and apply it to graphviz for generating visualization.

Handy_sqlite.py

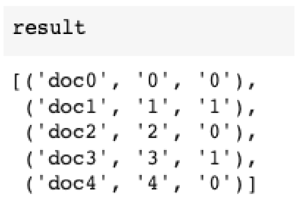


```
#Define table entities
doctor_entity = [('Name','TEXT'),('Depart_ID','TEXT'),('Doc_ID','TEXT')]

#create objects/tables
doctor = hg.sqlite_data_table(table_name = 'Doctor', table_entity = doctor_entity)

#convert to sql create table statement
doctor.create_table()

doctor.insert_table('dummy_data.csv','insert_doc.sql')
```

With the 'create_table' method(usage shown in the above image), the SQL script of creating the 'Doctor' table will be generated (as shown in the image on the right). With the ‘insert_table’ method (usage shown above) another SQL script will be created to input the data from a csv file into the database(.db) file.



```
CREATE TABLE Doctor(
Name TEXT,
Depart_ID TEXT,
Doc_ID TEXT);
```



```
result

[('doc0', '0', '0'),
 ('doc1', '1', '1'),
 ('doc2', '2', '0'),
 ('doc3', '3', '1'),
 ('doc4', '4', '0')]
```

The image on the left hand side is the fetch output of the demo code from the database (.db) file, after executing the SQL scripts created by the create_table() and insert_table() functions. The demo code for using ‘handy_sqlite’ module can be found on github repository ‘handy_sqlite_demo.py’

Discussion

The code in this package offers an effective way to work with SQLite databases using an object-oriented approach. With these methods, users can create and manage data tables, set and get primary and foreign keys, generate SQL statements for creating and inserting data, and export the data table to an SQL file for future use or sharing. Therefore, this package has a place in the software ecosystem as it offers an alternative to other SQLite database management packages that may need to be more straightforward. In addition, the object-oriented approach of the package makes it more intuitive to understand for users who may not be familiar with the underlying SQL language. In comparison to other related public libraries such as SQL academy, and the graphviz package in the official python website, our code still operates on a fundamental level in SQLite data management and visualization tasks. For example, the graphviz package in the python website allows users to create, customize, and render advanced graph visualization as it has more than eight modules in the graphing package, but our code can be used for simple table management and visualization tasks

Statement of Contribution

Jinhao Lu: Creating the github repository, writing and troubleshooting both the demo code and the main code in the project, contributing to the README file in the repository.
Anish Rao: Writing and troubleshooting the handy_sqlite and handy_graphviz modules and creating all the unit testing files for each module.
Tianbao Piao: Assist other members in the team in formulating project ideas, responsible for the write-up assignments: drafting the project report and help proofreading.

References

Graphviz https://graphviz.readthedocs.io/en/stable/manual.html