CSC3170 PROJECT

Option 3

Team 10:
李佳齐 120090545 万茜 119010289
张泽萱 120090674 徐康裕 120090133
周泽睿 120090533 樊天宇 120090311
孙鑫昊 120090597

CONTENTS

1 Project design

2 Functionalities

3 Project case show

4 Summary

01

PART ONE

Project design

Introduction

This project involves writing a miniature relational database management system (DBMS) that stores tables of data, where a table consists of some number of labeled columns of information.

This database supports simple query language for extracting information from tables and it will not consider efficiency and speed.

The database is implemented by python

Supported query language

create table <table name> (<column name>+ ,)
create table <table name> as <select clause>
load <name>;
store <table name>;
insert into <table name> values <literal>+
print <table name> ;
quit ;
exit ;
select <column name>+ , from <table name> <condition clause>
select <column name>+ , from <table name> , <table name> <condition clause>

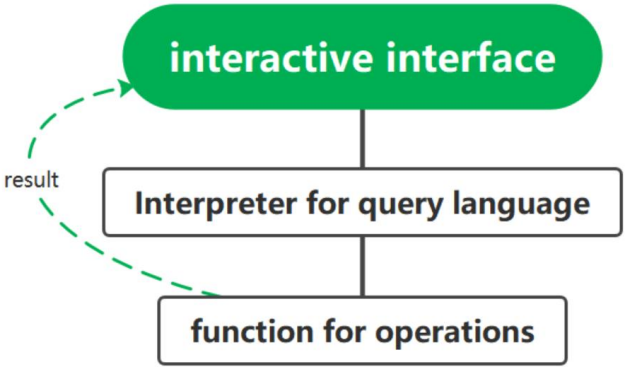
Structure of this database

This database consists of there main parts:
User interface: to read the instruction from users and show the output

Interpreter: to interpret the instruction and generate command, it will detect syntax error as well.

Functions for operation: to execute different type of commands and deal with error in database. It will finally return the result to user interface.

Data are stored in list and table are identified by its unique name



Information of code document

- **Command.py: class Command which is used to pass information from interpreter to functions**
- **Condition.py: class Condition which stores minimum unit of condition clause**
- **Database.py: class Database that includes functions for different queries:**
  - create table
  - create table as
  - load
  - store
  - insert
  - print
  - select
- **Interpreter.py: class Interpreter for interpreting command**
- **Main.py: user interface**
- **Row.py: class Row which is the smallest unit of information**
- **Table.py: class Table which is the structure of a table and it allows operation between tables**

02

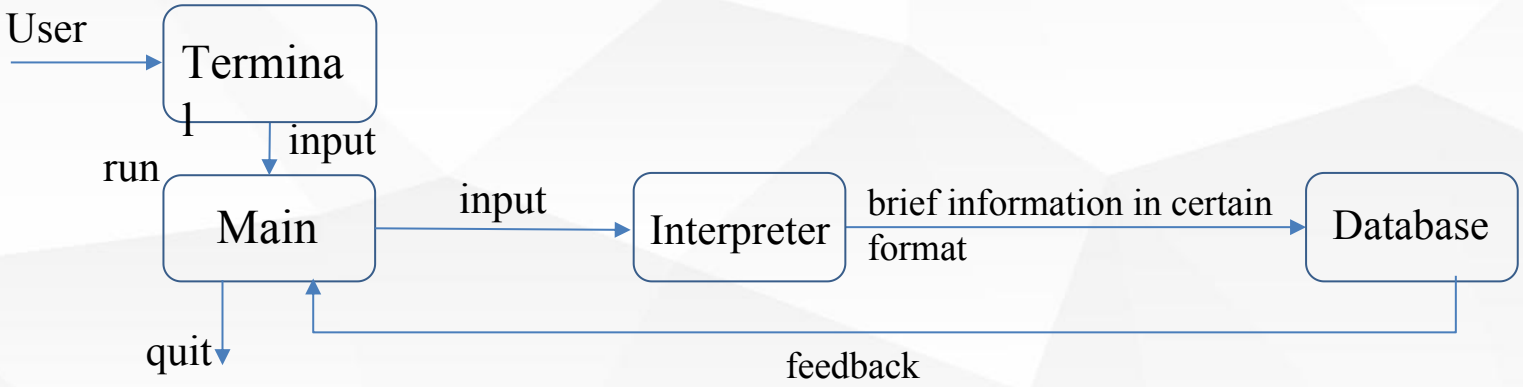
PART TWO

Functionalities

Main

Run the whole program:

- Read input from terminal;
- Send input to interpreter;
- Execute return object from interpreter in database;
- Judge whether quit or execute;



Interpreter

Function:

- fetch information from the SQL command
- detect SQL syntax error

Input: a string of a SQL command

Output: a “Command” class

```
'select a, b from c;'
```

```
class Command:
    type = "select"
    name = ['c']
    rows = []
    column = ['a', 'b']
    condition = []
```

Interpreter

Implementation

- detect command type

```
def typeDectect(command):
```

- decompose the command string according to its type

```
def select_interpreter(commandl, commands):
```

```
def insert_into_interpreter(commandl, commands):
```

- pack these functions together, create an interface for other trammates

```
def interpreter(instr):
```

- raise an exception if any error detected

Table

Table initiation:

“Table” class contains three attributes: name, column name, and data.

Join:

- If there is no element in “flag”, the tables do not have the same columns, then do outer join.
- If there is an element in “flag”, the tables have the same column, then do natural join.

Create Table

create table <table name> (<column name>+ ,)

Creates an empty table with the given name(replacing it if it is already loaded).

create table <table name> as <select clause>

Creates a table with the given name (replacing it if it is already loaded), whose columns and contents are those produced by calling select function.

Load & Store

load <name>;

- If the file exists, load data from the file name.db to create a table named table.
- If the file doesn't exist, output error information and stop loading.

store <table name>;

- If the table has been created or loaded, store the data from the table table name into the file table name.db.
- If the table doesn't exist, output error information and stop storing.

Insert

insert into <table name> values <literal>+, ;

- If the table has been created or loaded and the number of inserted data is valid, add a new row to the given table whose values are given by the list of literals. This command has no effect if there is already a row in the table with these values.
- If the table doesn't exists or the number of inserted data is invalid, output error information and stop inserting.

Print

print <table name> ;

- If the table has been created or loaded, print all rows of the table with the given name in certain format.
- If the table doesn't exists, output error information and stop printing.

Select

select<self, column, table, condition>;

- **select** <column name>$^{+}$, **from** <table name> <condition clause>
- **select** <column name>$^{+}$, **from** <table name> , <table name> <condition clause>
- If there is only one table name in the table list, loop through all tables in the database and compare the name with the table name in the table list. If a matching table name is found, the table is assigned to the *"ori_table"* variable.
- If there are two table names in the table list, a natural join operation will be performed.

03

PART THREE

Project case show

04

PART FOUR

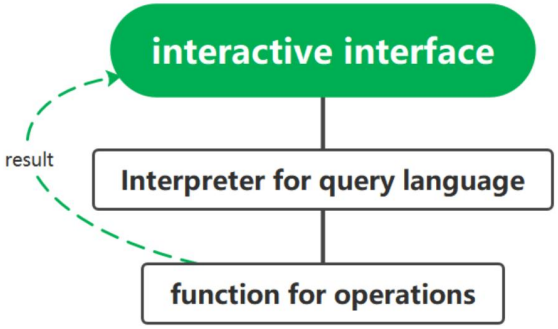
Summary

Summary

This is a DBMS implemented by python that support some simple query language.

It is decomposed into three main parts: User interface, Interpreter, Function for different queries.

It doesn’t consider speed and efficiency.



Promotion

Efficiency can be improved by applying some data structure to store the information such as hash and BST

To implement more functions for operations to support more complex query language

Design buffer memory and intermediate table for rollback and recovery

THANK YOU FOR WATCHING

Team 10:
李佳齐 120090545 万茜 119010289
张泽萱 120090674 徐康裕 120090133
周泽睿 120090533 樊天宇 120090311
孙鑫昊 120090597