11.12.2017

CENG113 Programming Basics

HOMEWORK #3 Database Management

In this homework, you will implement a (very) simple database management system for our custom database query language. Your program should ask for queries to execute until program termination.

In this section, semantics of the query language (i.e. permanent changes on database and console output) will be explained. This intuitive explanation is based on examples. For a formal specification of syntax and example scenarios, see Appendix A and B. Appendix C provides useful snippets and Appendix D explains evaluation criteria.

Create File

The query below will create a file to store some attributes (namely, “name”, “department” and “email”) of students and then print one of the given outputs below:

create file students with name,department,email

Output:

```
Corresponding file was successfully created.
```

or (if a file with such name already exists)

```
There was already such a file. It is removed and then created again.
```

or (if the query contains *id* as an attribute)

```
You cannot create a file with attribute 'id'.
```

Delete File

The query below will delete the corresponding file and then print one of the outputs given below:

delete file employee_information

Output:

```
Corresponding file was successfully deleted.
```

or (if such a file does not exist)

```
There is no such file.
```

Display Files

The query below will print such an output:

display files

Output (an example):

```
Number of files: 2
1) students: id,name,department,email
2) countries: id,name,GDP,continent
```

Add Line

The query below will add a line to the corresponding file and then print one of the outputs given below:

add Thimo_Adler,CENG,lorem@ipsum.com into students

Output (id may be different):

```
New line was successfully added to students with id = 1.
```

or (if there are missing or extra values in query)

```
Numbers of attributes do not match.
```

or (if such a file does not exist)

```
There is no such file.
```

Note: A special attribute named *id* is never explicitly mentioned in queries to create files, but it is automatically added to any file. When a line is added to a file, the corresponding *id* is not provided by user. Instead, a unique positive integer should be generated by your code and assigned to that line as *id*. Please note that i*d*s do not have to be consecutive, but they have to be unique.

Remove Lines

The query below will remove some lines from the corresponding file and print one of the outputs given below:

remove lines from countries where continent == South_America

Output (number of lines may be greater than or equal to 0):

```
2 lines were successfully removed.
```

or (if the attribute in condition does not exist in file)

```
Your query contains an unknown attribute.
```

or (if such a file does not exist)

```
There is no such file.
```

Modify Lines

The query below will modify some lines in the corresponding file and then print one of the outputs given below:

modify email in students as loremipsum@iyte.edu.tr where email == lorem@ipsum.com

Output (number of lines may be greater than or equal to 0):

```
0 lines were successfully modified.
```

or (if user wants to change id value, which is not allowed):

```
Id values cannot be changed.
```

or (if an attribute that is mentioned in query does not exist in file)

```
Your query contains an unknown attribute.
```

or (if such a file does not exist)

```
There is no such file.
```

Fetch Lines

The query below will fetch some lines from the corresponding file and then print one of the outputs given below:

fetch name,id,email from students where id != 2

Output (an example, note that columns are aligned):

```
Number of lines in file students: 4
Number of lines that hold the condition: 3
-------------------------------------------------------------
| name                        | id | email                    |
-------------------------------------------------------------
| Lorem_Ipsum                 | 1  | lorem.ipsum@example.com  |
| Dolor_Sit_Amet              | 5  | hello@dolorsitamet.info  |
| Consectetur_Adipiscing_Elit | 6  | 123456@iyte.edu.tr       |
-------------------------------------------------------------
```

or (if an attribute that is mentioned in query does not exist in file)

```
Your query contains an unknown attribute.
```

or (if such a file does not exist)

```
There is no such file.
```

Invalid Queries

If a query does not fit into formal definition of valid queries in Appendix A, then print the output:

```
Invalid query.
```

Program Termination

Program will terminate if the user enters *x* as input.

Persistency

There are 7 query types. Of the 7 query types, only 2 make no change on database (namely, 'display' and 'fetch'). If a query requires to make a change on database, make the change persistent. First, save the changes into the related file, and then print the information message to the console. This will guarantee that example scenarios will run the same way even if user restarts your program in the middle of a scenario.

File Format

You can decide your file extension and design content format as you wish as long as it is consistent. A sequence of arbitrary queries and a final version of corresponding file in an example format are given below. Note that output is ignored.

create file Players with name,team
add Cristiano_Ronaldo,Real_Madrid into Players
add Lionel_Messi,FC_Barcelona into Players

Players.hw3

```
id,name,team
1,Cristiano_Ronaldo,Real_Madrid
2,Lionel_Messi,FC_Barcelona
```

Appendix A: Syntax of Query Language

Specification of our query language in (simplified) Backus-Naur form is shown below. Note that vertical bars are used to provide alternatives.

Query ::= **Create** | **Delete** | **Display** | **Add** | **Remove** | **Modify** | **Fetch**

Create ::= create file **Identifier** with **Identifiers**
Delete ::= delete file **Identifier**
Display ::= display files
Add ::= add **Identifiers** into **Identifier**
Remove ::= remove lines from **Identifier** where **Identifier Operator Identifier**
Modify ::= modify **Identifier** in **Identifier** as **Identifier** where **Identifier Operator Identifier**
Fetch ::= fetch **Identifiers** from **Identifier** where **Identifier Operator Identifier**

Operator ::= ==|!=
Identifiers ::= One or more **Identifier** s connected with commas
Identifier ::= Any non-empty string that contains only English letters, digits, '_' (underscore), '.' (dot) and '@' (at)

Warning: Please mind space characters around operators. Note that queries are totally case-sensitive (e.g. "DISPLAY FILES" is invalid and "delete file BOOKS" will not remove "books"). Also note that space character is not allowed for an identifier (e.g. "add Tales from the Borderlands,2014 into Games" is an invalid query), and there is no space character between identifiers (e.g. "create file countries with name, GDP" is an invalid query).

Any query that does not fit into definition above should be counted as invalid. A valid query for each type is shown below:

Query Type	**Valid Query Example**
Create	create file employee.information with name,department,salary
Delete	delete file CENG_Students
Display	display files
Add	add Cristiano_Ronaldo,Real_Madrid into Players
Remove	remove lines from movies where id != -1
Modify	modify type in publications as conference_paper where type == conference
Fetch	fetch id,salary from employees where department == software_development

Note: Each condition consists of an attribute name and a value. The attribute name is always at the left-hand side and the value is at the right-hand side.

Assumption: Special English words that are mentioned in query definitions are keywords (e.g. "create", "where", "in", etc.). Identifiers may contain keywords (e.g. "Of_Mice_and_Man" that contains "and" and "Geoffrey_Hinton" that contains "into" are valid identifiers). However, for the sake of simplicity, you can safely assume that no identifier will be equal to a keyword (e.g. user will not enter "files", "lines" and "as" as identifiers).

Appendix B: An Example Scenario

```
What is your query?
create file books with ISBN,title,genre,author,publisher
Corresponding file was successfully created.

What is your query?
create file books with ISBN,title,genre,author,publisher,year
There was already such a file. It is removed and then created again.

What is your query?
fetch id,title,author from books where id == 1
Number of lines in file students: 0
Number of lines that hold the condition: 0
-----------------------
| id | title | author |
-----------------------

What is your query?
add 1943138427,Animal_Farm,Dystopian,George_Orwell,Penguin_UK,1946 into books
New line was successfully added to books with id = 1.

What is your query?
add 0679722769,Ulysses,Fiction,James_Joyce,Dover,1922 into books
New line was successfully added to books with id = 2.

What is your query?
add 9780060850524,Brave_New_World,Dystopian,Aldous_Huxley,Harper into books
Numbers of attributes do not match.

What is your query?
fetch id,title,author from books where id == 1
Number of lines in file students: 2
Number of lines that hold the condition: 1
-----------------------------------
| id | title       | author        |
-----------------------------------
| 1  | Animal_Farm | George_Orwell |
-----------------------------------

What is your query?
create file movies with title,genre,director,minutes,year
Corresponding file was successfully created.

What is your query?
add Yesterday,Help,The_Beatles,Rock,1966 into songs
There is no such file.

What is your query?
add Death_Note,Adventure,Adam_Wingard,101,2017 into movies
New line was successfully added to movies with id = 1.

What is your query?
add My_Neighbor_Totoro,Anime,Hayao_Miyazaki,86,1988 into movies
New line was successfully added to movies with id = 2.
```

```
What is your query?
display files
Number of files: 2
1) books: id,ISBN,title,genre,author,publisher
2) movies: id,title,genre,director,minutes,year

What is your query?
create file songs with title,album,artist,genre,year
Corresponding file was successfully created.

What is your query?
display files
Number of files: 3
1) books: id,ISBN,title,genre,author,publisher
2) movies: id,title,genre,director,minutes,year
3) songs: id,title,album,artist,genre,year

What is your query?
add Yesterday,Help,The_Beatles,Rock,1966 into songs
New line was successfully added to songs with id = 1.

What is your query?
add Sea_Of_Monsters,Yellow_Submarine,The_Beatles,Rock,1969 into songs
New line was successfully added to songs with id = 2.

What is your query?
add Time,The_Dark_Side_Of_The_Moon,Pink_Floyd,Rock,1973 into songs
New line was successfully added to songs with id = 3.

What is your query?
remove lines from songs where artist == The_Beatles
2 lines were successfully removed.

What is your query?
remove lines from books where artist == James_Joyce
Your query contains an unknown attribute.

What is your query?
delete file songs
Corresponding file was successfully deleted.

What is your query?
remove lines from songs where artist == Pink_Floyd
There is no such file.

What is your query?
modify album in songs as Greatest_Hits where title == Yesterday
There is no such file.

What is your query?
modify publisher in books as Vintage where titl == Ulysses
Your query contains an unknown attribute.

What is your query?
modify publisher in books as Vintage where title == Ulysses
1 lines were successfully modified.
```

```
What is your query?
modify id in movies as 5 where title == Death_Note
Id values cannot be changed.

What is your query?
add 9750718533,Nineteen_Eighty_Four,Dystopian,George_Orwell,Penguin,1949 into books
New line was successfully added to books with id = 3.

What is your query?
fetch id,title from books where author == George_Orwell
Number of lines in file students: 3
Number of lines that hold the condition: 2
-----------------------------
| id | title                |
-----------------------------
| 1  | Animal_Farm          |
| 3  | Nineteen_Eighty_Four |
-----------------------------

What is your query?
Display files
Invalid query.

What is your query?
x
```

Appendix C: Some Useful Snippets

The snippet below will retrieve names of files that exist in your working directory. You can later filter them according to their extensions.

```
import os
file_names = os.listdir()
```

The following snippet will delete the mentioned file in your working directory. Furthermore, you should check if such a file really exists (using os.listdir()) before executing the remove function in order to prevent possible errors.

```
import os
file_name = "Mona Lisa.jpg"
os.remove(file_name)
```

Note: Importing os (operating systems) module for once is necessary and sufficient.

Appendix D: Submission Rules and Evaluation Criteria

Submission Rules

- You should submit your assignments through CMS until due date.
- Your homework should be named as CENG113_hw3_studentID.py (e.g. CENG113_hw3_123456789.py).
- Write your student ID as a comment at the beginning of your code.

Note: Violation of any submission rule may end up with point deduction up to 100 points. Any kind of cheating including teamwork will be penalized. Your program will be tested on Linux.

Evaluation Criteria

(70 points) **Performance on test cases:** correctness of console output with respect to functionality and style for several independent scenarios including given scenario in Appendix B, short scenarios, single-query scenarios, invalid queries, and program termination between queries.

(15 points) **Code readability:** meaningful identifiers (names of variables, functions, etc.) with a consistent style, comments (when, and only when necessary), indentation and consistent use of spaces, separation of main program and function definitions, etc.

(15 points) **Modular programming:** control abstraction (functions for subprograms and sub-subprograms when necessary, relatively short and simple main program, etc.), reusable code, absence of code repetition, etc.