TRADITIONAL FILE SYSTEM

CSE 4308
Database Management Systems Lab

Hasin Mahtab
210042174
Department of CSE
B.Sc in Software Engineering

August 13, 2023

Contents

1 Task 01

The traditional file system was used to store information prior to the development of databases. And it was no easy task to maintain the information in that manner. We will see what difficulties we face while storing data in simple text files.

Solution 1.1. Print average GPA from .txt file

The database consists of two files, namely 'studentInfo.txt' and 'grades.txt'. Each row of the studentInfo.txt file contains Student ID, Name, Age, Blood Group, and Department of a student. And each row of the grades.txt file contains the Student ID, GPA, and the Semester in which that GPA was achieved.

In the first task, we will see how we can iterate through all the lines in the text file to find the GPA points of each student and also keep a count of all the lines or students we have in the list to find the average GPA of the students in the grades.txt file.

```python
def calculateAverageGpa(gradesFile):

        content = open(gradesFile, "r")
        lines = content.readlines()
        content.close()

        totalGpa = 0
        for line in lines:
                totalGpa += float(line.split(";")[1])

        average = "{:.2f}".format(totalGpa / len(lines))
        print("Average GPA is: ", average)


if __name__ == "__main__":
        calculateAverageGpa("grades.txt")
```

Here, we read the text file using the "r" permission and then save all the lines in a list. We iterate over the list for each line and then we separate the GPA using string split method and giving the parameter as ";". For each GPA we add it to the toalGpa variable and then divide it with the number of lines we have in the list and we get the average GPA.

2 Task 02

Inserting valid data is another important aspect when it comes to maintaining databases and information. In text files, there are no built-in ways to INSERT data neither is there a way to validate that data. So we need to check conditions before inserting data.

Solution 2.1. INSERT student info to .txt file

Here, we take input from the user and then check those values against our set conditions to return true or false before inserting the values. To insert the value we need to append them in the text file as a new line.

```python
def INSERT_DATA(gradeInfoFile):
        studentID = str(input("Enter your Student ID: "))
        studentGPA = round(float(input("Enter GPA: ")), 2)
        semester = int(input("Enter Semester: "))

        isValid = Check_Data(studentID, studentGPA, semester)

        if isValid == False:
                print("Invalid Data")
                return
        else:
                toAppend = studentID + ";" + str(studentGPA)
                + ";" + str(semester)
                append_new_line(gradeInfoFile, toAppend)
                print("Data Inserted Successfully")


def append_new_line(fileName, textToAppend):
        with open(fileName, "a+") as file_object:
        file_object.seek(0)
        data = file_object.read(100)

        if len(data) > 0:
                file_object.write("\n")
                file_object.write(textToAppend)
```

```python
def Check_Data(stuID, GPA, Sem):
        if GPA < 2.50 or GPA > 4.00:
                return False
        if Sem < 1 or Sem > 8:
                return False
        if len(stuID) >= 11:
                return False

        return True


if __name__ == "__main__":
        INSERT_DATA("grades.txt")
```

Difficulties

- Setting the validity checking conditions for inserting the data.
- Have to manually insert line breaks.

3 Task 03

We need to take the ID as input and then print the name of that student. We also need to print the semester in which he/she has the lowest grade.

Solution 3.1. Return name and semester from .txt file

First we need to search for the ID in students information file and if we can find it, the we separate name and print it. Once we find the name, we can now check for the ID in grades file. From the grades file, we need to save all the grade corresponding to the ID and save them in a list, then we need to find the lowest and then once again iterate though the file to find the ID and lowest grade in the same line and then return the semester number in which that student has the lowest grade.

```python
def QueryStudentInfo(studentInfoFile, gradesFile):
        studentID = input("Enter student ID: ")
        ShowName(studentID, studentInfoFile)
        ShowGrades(studentID, gradesFile)



def ShowName(studentID, File):
        studentInfo = open(File, "r")
        studentFound = False
        studentName = ""

        for line in studentInfo:
                if line.startswith(studentID):
                        studentFound = True
                        studentName = line.split(";")[1]
                        break
                else:
                        studentFound = False
        studentInfo.close()

        if studentFound:
                print(studentName)
        else:
                print("Student not found")
```

```
def ShowGrades(studentID, File):
        grades = open(File, "r")
        studentGrades = []
        lowestGrade = 0.0
        lowSemester = ""

        for line in grades:
                if line.startswith(studentID):
                        studentGrades.append(line.split(";")[1])

        if studentGrades != []:
                for grade in studentGrades:
                        lowestGrade = float(min(studentGrades))
                        print("Grades: ", studentGrades)
                        print("Lowest grade: ", lowestGrade)

        else:
                print("Student not found")

        grades.seek(0)
        for line in grades:
                stuID = line.split(";")[0]
                grade = float(line.split(";")[1])

                if stuID == studentID and grade == lowestGrade:
                        lowSemester = line.split(";")[2]
                        break

        if lowSemester != None:
                print("Semester with lowest grade: ", lowSemester)
        else:
                print("Student not found")

        grades.close()


if __name__ == "__main__":

        QueryStudentInfo("studentInfo.txt", "grades.txt")
```