NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Course Name

Database management system implementation technology experiment

Topic

数据库管理系统实现技术实验

Database Management System Laboratory report

Prepared For

Qin Xiaolin

School of Computer Science and Technology

Prepared By

MD.OHIDUL BARIK

SL1816013

Submission Date

27th june, 2019

1. Introduction

Databases rank among the most significant structural elements of the current modern era. Lying in the basis of the majority of the data based activities, such as business and research, databases serve a special mission - to provide a well-organized mechanism for data manipulation. The database approach in current day offer a quick and automated way for the information to be stored, managed, deleted or retrieved. Databases' powerful set of capabilities has determined the grow in current industry such as AI, which has opened a new page in the evolution history of the world.

Database refers to a collection of electronic records that could be processed to produce useful information. The data can be accessed, modified, managed, controlled and organized to perform various data-processing operations. Database Management System (DBMS) refers to the technology solution used to optimize and manage the storage and retrieval of data from databases. DBMS offers a systematic approach to manage databases via an interface for users as well as workloads accessing the databases via apps.

In this project, I present a DBMS program which correspond to the given task listed in Chapter 2. The structure of this project is: Chapter 1 is introduction, Chapter 2 is the selected task from the list of tasks given by the class instructor, Chapter 3 is the system setting which being used for developing the program, Chapter 4 is the resulted program along with brief explanation of the functionality, and Chapter 5 is the conclusion.

2. Selected Task (题目)

题目 1：基于主存的 DBMS 存储管理机制

1．功能

⑴内存中装入系统、数据库、应用程序和工作空间。

⑵内存由 DBMS 管理，不由操作系统控制。

⑶装入机制（外存 DB 输入到内存）。

⑷写出机制。

⑸DB 状态。

2．测试方法

⑴外存上建立长度不同的 10 个文件 f1、f2、f3……f10。

⑵全部装入内存。

⑶对内存中的 fi 进行查询、插入、删除、更新操作。

⑷把内存中的 f1~f10 写到外存 file.db 中，包括内存使用情况信息。

⑸装入 file.db 对 fi 进行第 3 步操作。

3. System Program Setting

The program is written in Java 1.8 using MySQL 5.7.26 database, in Ubuntu 18.04 operating system. The program is written in Eclipse IDE, and the submitted program is an Eclipse Java Project. All external libraries, excluding JRE or JDK, are included in the file, including MySQL connector to Java (JDBC) and Apache's Commons IO.

4. The Resulted Program

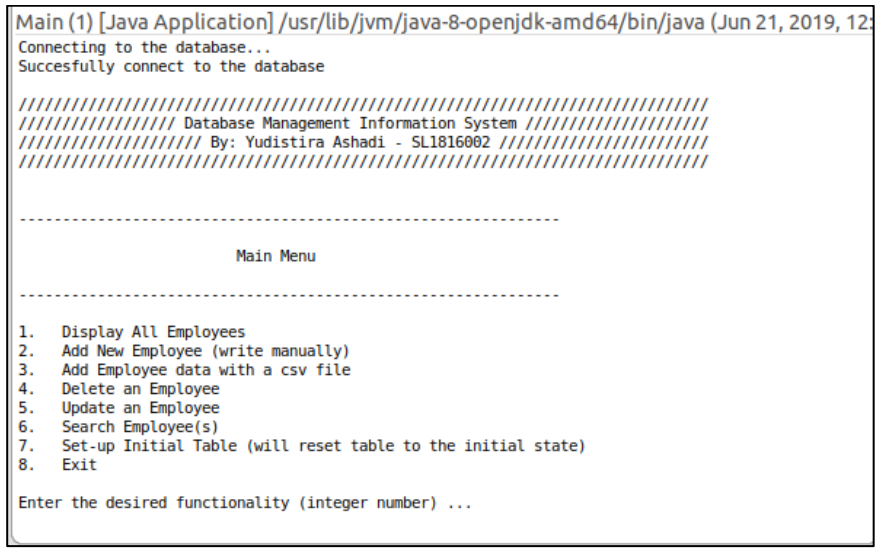


Figure 1. Main menu shown if connection to the database successfully established the first time the user open the program.

The resulted program is a DBMS program for an employee management system. the resulted program, or the program as I will state it for the rest of the report, able to display, add, delete, update, and search for an employee or employees. The program is connected with MySQL database, but can easily configured for other database framework, such as Oracle or SQLite, as long as it is supported by JDBC connector. The program is built in based on the selected task from the list of tasks by the class instructor, which can be found in Chapter 2.

When the first time opening the program, it will first try to connect to the database stated in the source code. When connection can not be established because of reasons such as denied database user verification or any other error, the program will printed the error message and will be immediately prompted to solve the error. When no error occurs and the connection can be established, the Command Line Interface (CLI) will show the interface exactly like Figure 1.

In the program's main menu, there are listed 8 menu options, or modules as I will state it for the rest of the report, that the user can select by typing the number of the module when prompted then press enter. The listed modules are:

(1) Display all employees, to display all employees data listed in the database;

(2) Add new employee (write manually), to insert a new employee by writing it in the CLI;

(3) Add employee data with a comma separated file (csv) file, to insert a csv file path consisting of employee data that wished to insert to the database;

(4) Delete an employee, to delete an employee data based on their ID;

(5) Update an employee, to update an employee data based on their ID;

(6) Search employee(s), to search employee or employees based on ID, Last Name, First Name, Email, Department, or Salary;

(7) Set-up initial table (will reset table to the initial state), to reset, or set if it has not been set before, the schema for the required table and insert 10 initial employees data;

(8) Exit, to exit the program.

Before opening the program, the user is advised to first set up the database credential (user's ID and password), database URL, database's name, and file path for the 10 required input files. The required input files are the needed 10 files consist of different length or size of data for testing the system, as stated in Chapter 2.

Upon the first time opening the program, the user is advised to first select module (7) to set-up the initial table and insert initial 10 employees. This will let the program to run *table-setup.sql* in data folder. Unless that file is already run or the table has been set up, I strongly denied against selecting the module (7) for the first module to be selected. The CLI upon selecting module (7) is shown in Figure 2. After almost any module, the user will be prompted to press enter to go back to the main menu.

Module (1) is used to display all employees data from the database. The user can use module (1) to check the employees data after adding changes to the database, such as adding new employee or deleting an employee. Figure 3 shows the result of selecting module (1) after re-setting the table with module (7). Notice that the figure only shows 4 employees data. In the real program, the CLI can be scrolled upward to shows the other employees data. The data is shown incrementally based on the employee ID number. The figure also shows that in the CLI, is shown in total the database have data of 10 employees, which correspond to the 10 initial employees inserted from module (7).

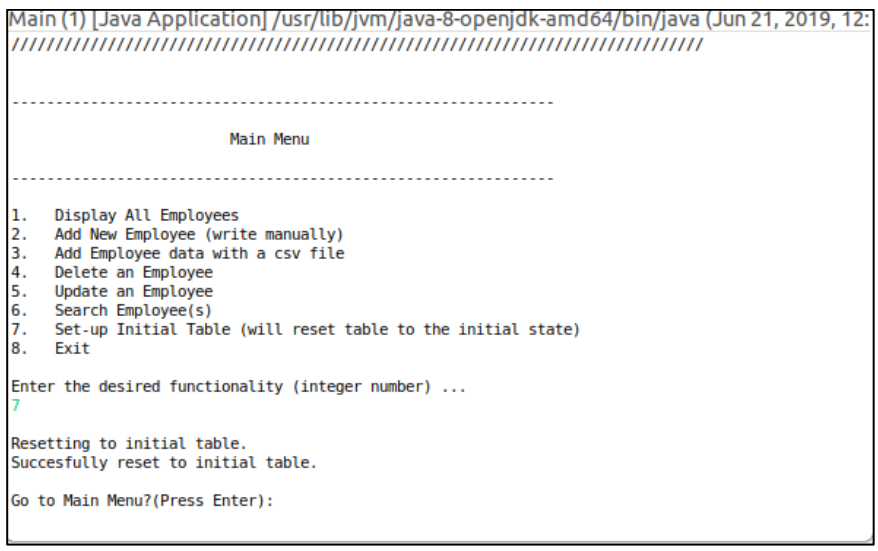
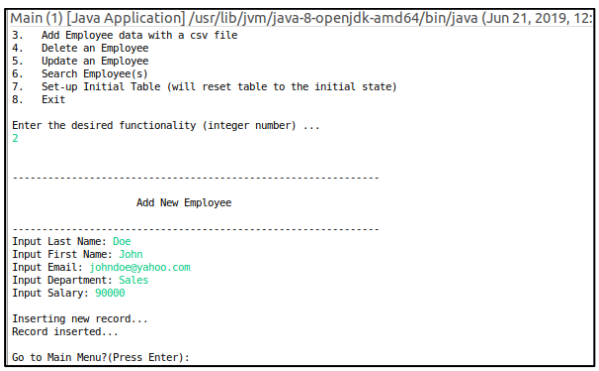


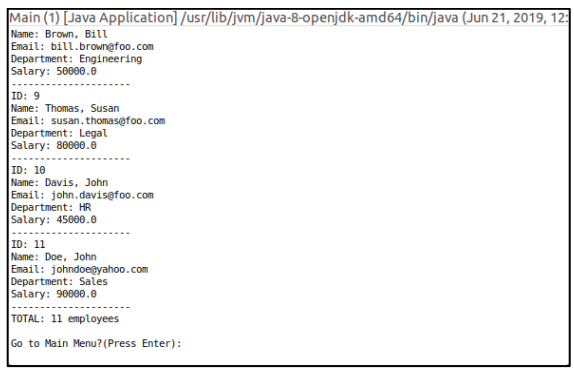
Figure 2. The CLI after selecting module (7) in the main menu.

Figure 3. The result of selecting module (1) after re-setting the table with module (7).

In module (2), the user can input a new employee data, which have to include their last name, first name, email, department, and salary. The data type are all *string*, except salary which is *double*. All the employee data have to be inserted respectively. Figure 4 shows the CLI and the state of the database after inserting a new employee data in module (2). The employee ID number can not be inserted explicitly, as the ID number were set to increment automatically for the newly inserted employee.



(a) The insertion process and right after insertion of new employee data.



(b) The result of module (1) after insertion a new employee data in Figure 4.a.

Figure 4. The CLI after inserting a new employee data in module (2).

Module (3) can be used to insert a csv file containing the employee data wished to be inserted into the database. This module is used to insert 10 required input files with different data length into the program, as requested in Chapter 2. Figure 5 shows the content of input1.csv, one of the required input files. The first line consist of the csv header, which only used as the data description. The required input file names, including their size are:

(1) input1.csv, 10 employees;

(2) input2.csv, 30 employees;

(3) input3.csv, 50 employees;

(4) Input4.csv, 80 employees;

(5) Input5.csv, 120 employees;

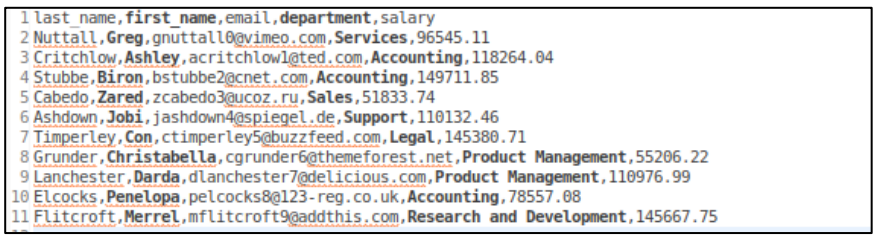
(6) Input6.csv, 170 employees;

(7) Input7.csv, 250 employees;

(8) Input8.csv, 390 employees;

(9) Input9.csv, 540 employees;

(10) Input10.csv, 800 employees.



```
last_name,first_name,email,department,salary
Nuttall,Greg,gnuttall0@vimeo.com,Services,96545.11
Critchlow,Ashley,acritchlow1@ted.com,Accounting,118264.04
Stubbe,Biron,bstubbe2@cnet.com,Accounting,149711.85
Cabedo,Zared,zcabedo3@ucoz.ru,Sales,51833.74
Ashdown,Jobi,jashdown4@spiegel.de,Support,110132.46
Timperley,Con,ctimperley5@buzzfeed.com,Legal,145380.71
Grunder,Christabella,cgrunder6@themeforest.net,Product Management,55206.22
Lanchester,Darda,dlanchester7@delicious.com,Product Management,110976.99
Elcocks,Penelopa,pelcocks8@123-reg.co.uk,Accounting,78557.08
Flitcroft,Merrel,mflitcroft9@addthis.com,Research and Development,145667.75
```

Figure 5. The content of input1.csv.

After the selection of module (3), the user will be prompted to select 1 of 2 available options, i.e. to insert a file or insert input1 to input10 continuously. Figure 6.a shows the CLI after selecting module (3) option 1, which I will not show the example of inserting a file. The path inserted can be full path, i.e., starting from the root of the user computer, or relative path, i.e., starting from the *src* folder of the resulted program. Figure 6.b shows the CLI after selection module (3) option 2. The figure shows that all the record are inserted successfully, which the user can checked with module (1), which shown in Figure 6.c.

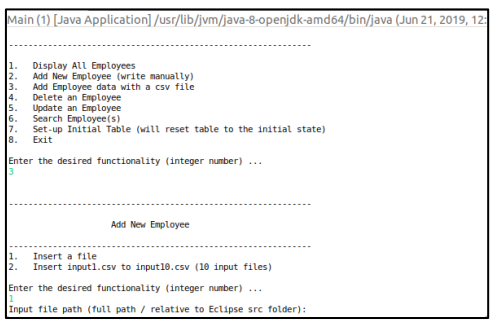
Module (4) is used to delete an employee from the database. The deletion is based on the employee ID. Figure 7.a shows the process of deletion in module (4). After selection module (4) in main menu, the user will be prompted to insert the ID of the employee which wished to be deleted. In the figure, it is shown that the deleted employee ID is ID 2451. The updated employee data can be checked using module (1), which is shown in Figure 7.b.

Module (5) is used to to update an employee. Figure 8.a shows the process of the update in module (5). The process starts with the insertion of the employee by employee ID which has to be inserted by the user. If the user doesn't exist, the program will thrown an error and the user will be prompted to insert another employee ID. If the inserted employee ID is exist, the user will be prompted the employee new information including last name, first name, email, department, and salary, respectively. The result after the update in Figure 8.a is shown in Figure 8.b.

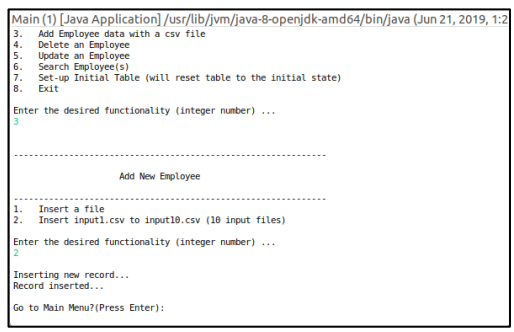
Module (6) is used to search for employee(s). Figure 9.a shows the search condition available in the module, which includes ID, last name, first name, email, department, and salary. Most of them are quite straightforward, which the program do the search based on exact input. For example, the search using last name condition, requires the user to input the last name that the user wants to search in case sensitive manner. After the insertion, the program will only give result with exactly the same last name. This condition is similar to all the conditions, except for salary. In salary, the user have 2 options, which are smaller than or bigger than. With smaller than option, the user can input a certain number of salary, then the program will query all employee which have the smaller than the number inputted. The same apply for bigger than option, but instead of the program querying the smaller than salary, it query the bigger than salary. Figure 9.c shows the CLI of the program for aforementioned behaviour. Figure 9.c also shows that the module (6) can accept more than one search condition. In the picture, I shows a search using 2 conditions, but if the user wants more conditions to be applied, there is no limit of search condition to be used. Figure 9.d shows the result after the multi-conditions search in Figure 9.c.

Module (7) is used to reset the current database and setup the table. When module (7) is selected, the program will dump all available data in current state of the database. It will create the table schema from zero, then refill the table with the initial 10 employees in the initial state. The process of resetting the table can be seen in Figure 10.b. The result can be seen in Figure 10.c, which the previous state of the database and the employee data can be seen in Figure 10.a as comparison.

Module (8) is used to exit the program gracefully. Although there is a module specially built for exiting the program, closing the program window by pressing the red X button is also can be used to exiting the program.



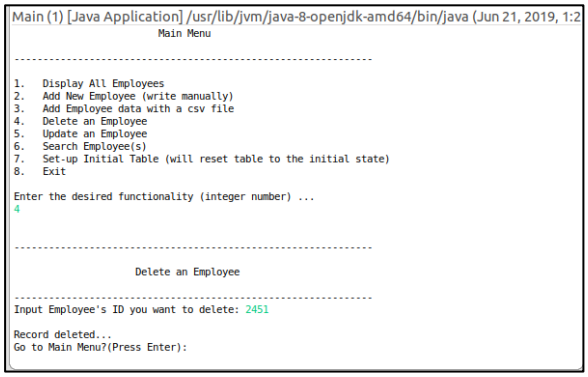
(a) The options available in module (3) and the prompted message after selecting option 1.



(b) The CLI after selecting option 2.

(c) The result of module (1) after insertion of new data in Figure 6.b.

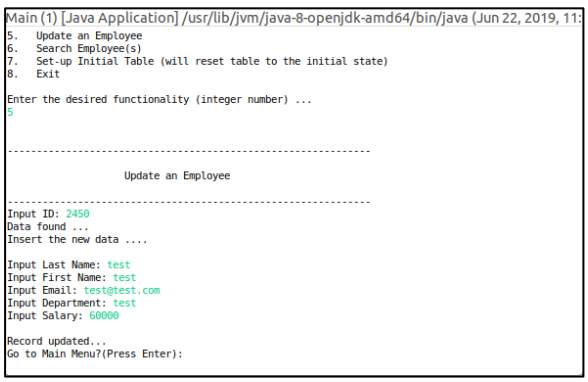
Figure 6. The CLI in module (3).



(a) The process of deletion in module (4).

(b) The result of module (1) after deletion of an employee data in Figure 7.b.

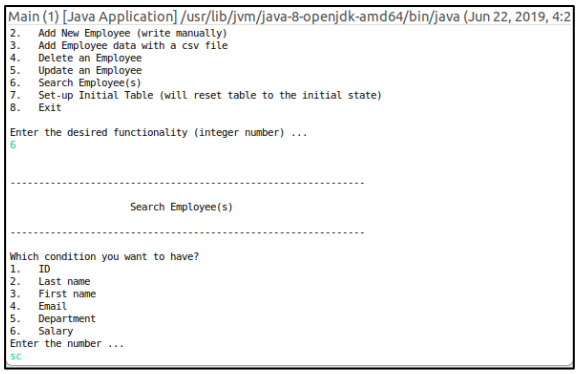
Figure 7. The CLI in module (4).



(a) The process of the update in module (5).

(b) The result of module (1) after the update in Figure 8.a.

Figure 8. The CLI in module (5).



(a) The options menu in module (6).

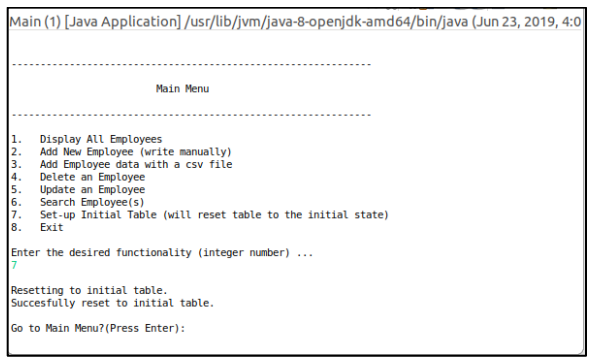
(b) The process and result of the search using 1 search condition.

(c) The process of the search using multiple (2) search conditions.

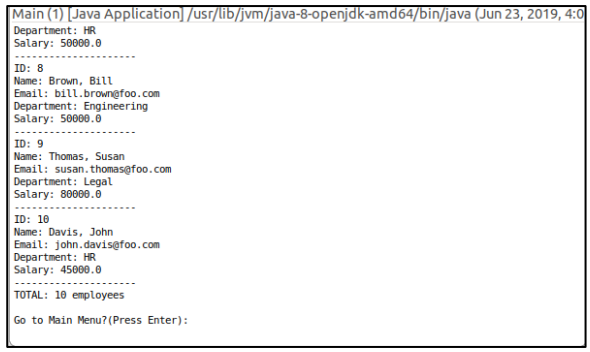
(d) The result of the search in Figure 9.c.

Figure 9. The CLI in module (6).

(a) The employee data before the selection of module (7).



(b) The process of data setup and resetting.



(c) The employee data after Figure 10.b.

Figure 10. The CLI in module (7).

5. Conclusion

In this report, I have presented the program based on the given task by the class instructor. The selected task can be seen in Chapter 2. The task is basically to build a DBMS program which corresponds to the requirement listed in Chapter 2. I have built a DBMS program to manage employee data, including their ID, last name, first name, email, department, and salary. The resulted program have many functionality, including displaying all employee data, add an employee or employees, delete an employee, update an employee, search for an employee or employees, and setup or reset the database to the initial state. I hope the resulted program is satisfy what the class instructor wants, as well as corresponds to the requirements.