

CSCI3170 (2018-2019 1st term)

Introduction to Database Systems

Project– Ridesharing System

Group Registration Deadline: 23:59 29th September 2018

Phase 1 Deadline: 23:59 12th October 2018

Phase 2 Deadline: 23:59 23rd November 2018

1. Introduction

You are required to implement a system for a ridesharing startup company for managing data relevant to ridesharing, e.g. drivers, passengers, vehicles, orders and trips. The system should provide an interactive interface to system administrators, passengers, and drivers.

This project is divided into 2 phases:

In phase 1, you are required to design the database for the system (including an ER-diagram and a relational schema). A suggested solution will be provided after phase 1. You are required to use the suggested solution to complete phase 2.

In phase 2, you are required to implement the system as a Java command-line program.

Our tutors will give tutorials on how to connect to a MySQL database system with JDBC API and deploy your work on the required platform.

This is a group project and each group should have at most 3 members. ONLY one copy of solution is required for each group. Please fill out the group registration form in Blackboard before the group registration deadline.

2. Milestones

Preparation

- Read the document thoroughly and make sure you understand all the assumptions and regulations stated in Section 4.

Phase 1 (20 %)

- According to the data specifications in Section 3, design an ER-diagram and transform it into a relational schema without any redundant fields and tables.

Phase 2 (80 %)

- According to the suggested solution of phase 1, implement a Java application that fulfills all requirements stated in Section 5.
- Debug your system with different datasets and user inputs.
- Write a readme file to describe the compilation and deployment of your system.

3. Data Files Specification

Every data file has a Unix line ending (i.e. the newline character is “\n”) and are encoded in utf-8. The system should read every record stored in each file and put them into appropriate tables of the provided MySQL DBMS via JDBC API. There are 4 input files in total listed in the subsections. Each line of each input file is a sequence of attributes delimited by a comma (,). The definition of each attribute in each input file is defined in the corresponding subsections. The order of the attributes within a line of each input file follows that of the attribute in the corresponding subsection. A sample data set will be provided after the deadline of Phase 1.

3.1. Drivers – drivers.csv

Attribute Name	Format	Description
ID	A positive integer	A unique identifier for the driver.
Name	A non-empty string with at most 30 characters	The name of the driver.
Vehicle ID	A non-empty string with 6 characters	The ID of the vehicle which belongs to the driver.

3.2. Vehicles – vehicles.csv

Attribute Name	Format	Description
ID	A non-empty string with 6 characters	The license plate number of the vehicle, which uniquely identifies a vehicle.
Model	A non-empty string with at most 30 characters	The model of the car, e.g., Toyota Prius, Tesla Model X.
Model Year	A positive integer with 4 digits	The full year the vehicle is produced.
Seats	A positive integer	The number of seats, excluding the driver seat, of the vehicle.

3.3. Passengers – passengers.csv

Attribute Name	Format	Description
ID	A positive integer	A unique identifier for the passenger.
Name	A non-empty string with at most 30 characters	The name of the passenger.

3.4. Trips – trips.csv

Attribute Name	Format	Description
ID	A positive integer	A unique identifier for the trip.
Driver ID	A positive integer	The ID of the driver of the trip.
Passenger ID	A positive integer	The ID of the passenger of the trip.
Start	A time (format see 4.1)	The time when the trip starts.
End	A time (format see 4.1)	The time when the trip ends.
Fee	A positive integer	The fee of the trip.
Rating	A positive integer	The rating of this trip given by the passenger. A rating of 0 means the trip has not been rated yet.

4. Assumptions

4.1. System

- All numerical values will not be larger than the maximum integer value that can be handled by Java.
- Every time input should follow the format “YYYY-MM-DD HH:mm:ss”, e.g., “2018-08-10 23:08:53”, regardless of time zone.
- Every date input should follow the format “YYYY-MM-DD”, e.g. “2018-07-15”.
- There is no duplicate row in any input files.
- Any user inputs to the system are correct in format only.
- Every data file is correct in format and content.

4.2. Drivers

- Every driver should be uniquely identified by his/her ID.
- Every driver should have only 1 vehicle.

4.3. Vehicles

- Every vehicle should be uniquely identified by its ID, which is also its license plate number.
- Each vehicle should have 3 to 7 seats excluding the driver seat.
- Every vehicle should belong to only 1 driver.
- Each vehicle should have a model year between 2010 and 2018.

4.4. Passengers

- Every passenger should be uniquely identified by his/her ID.

4.5. Requests

- Every request should be uniquely identified by its ID.
- Each request should have 1 to 8 passengers.
- A request should be marked taken once it is taken by a driver.
- An open request is a request which is not taken by any driver yet.
- A passenger should not be allowed to place request if there is an open request by him/her.
- The model criterion of every request should be of length at maximum 30 characters.

4.6. Trips

- Every trip should be uniquely identified by its ID.
- A trip should be created once a driver took a request. Therefore, there may be trips which are unfinished.
- There may be trips without a rating as passengers can choose not to give a rating to a trip.

5. System Function Requirements

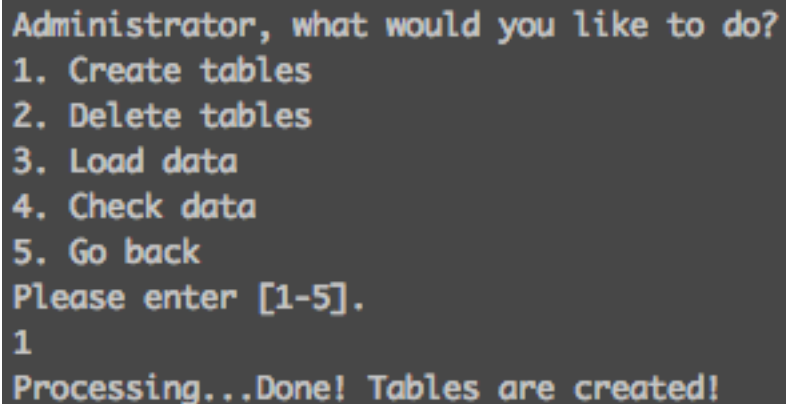
You are required to write a simple command line application in Java. After performing an operation, the program should display the last appeared menu. The Java program should provide the following functions:

5.1. System Administrator

The system should let administrators to perform the following operations:

- **Create tables**

An administrator can create all the tables in the database based on the given relational schema.

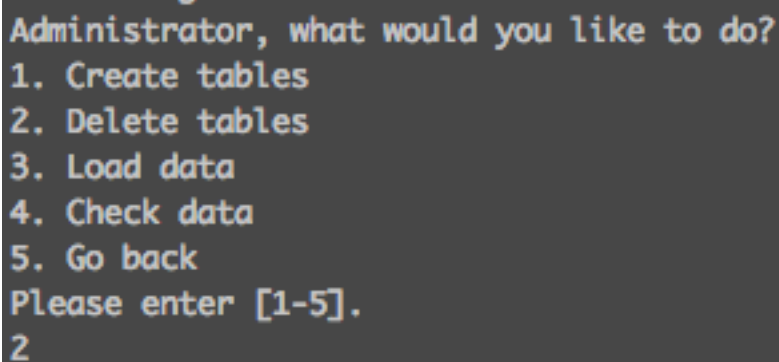
A screenshot of a terminal window with a dark background and light-colored text. The text shows a prompt 'Administrator, what would you like to do?' followed by a numbered list of five options: '1. Create tables', '2. Delete tables', '3. Load data', '4. Check data', and '5. Go back'. Below the list is the instruction 'Please enter [1-5].'. The number '1' is entered as input. The final line of output is 'Processing...Done! Tables are created!'.

```
Administrator, what would you like to do?  
1. Create tables  
2. Delete tables  
3. Load data  
4. Check data  
5. Go back  
Please enter [1-5].  
1  
Processing...Done! Tables are created!
```

Figure 1: Example output of creating tables.

- **Delete tables**

An administrator can delete all tables in the database created based on the given relational schema.

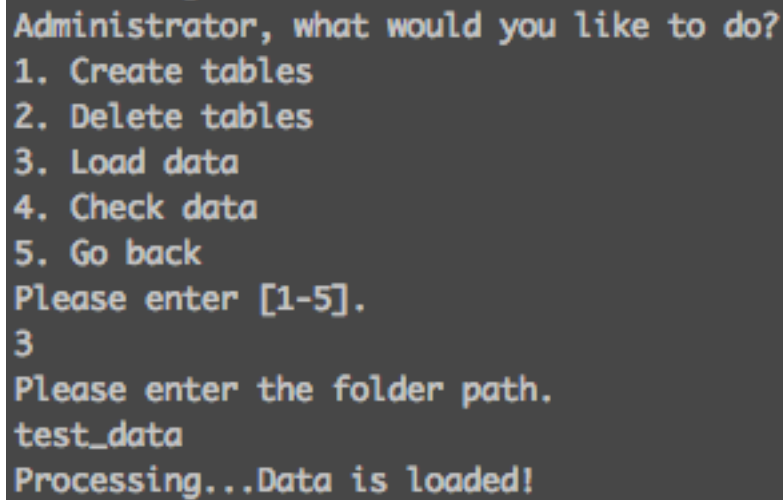
A screenshot of a terminal window with a dark background and light-colored text. The text shows a prompt 'Administrator, what would you like to do?' followed by a numbered list of five options: '1. Create tables', '2. Delete tables', '3. Load data', '4. Check data', and '5. Go back'. Below the list is the instruction 'Please enter [1-5].'. The number '2' is entered as input.

```
Administrator, what would you like to do?  
1. Create tables  
2. Delete tables  
3. Load data  
4. Check data  
5. Go back  
Please enter [1-5].  
2
```

Figure 2: Example output of deleting tables.

- **Load data**

An administrator can load the system with data. The system should let the administrator enter the path of the folder containing the data files, then read each data file and load it into a table in the database. You may assume the folder contains all 4 data files, namely drivers.csv, vehicles.csv, passengers.csv, and trips.csv. See section 3 for data files specifications.

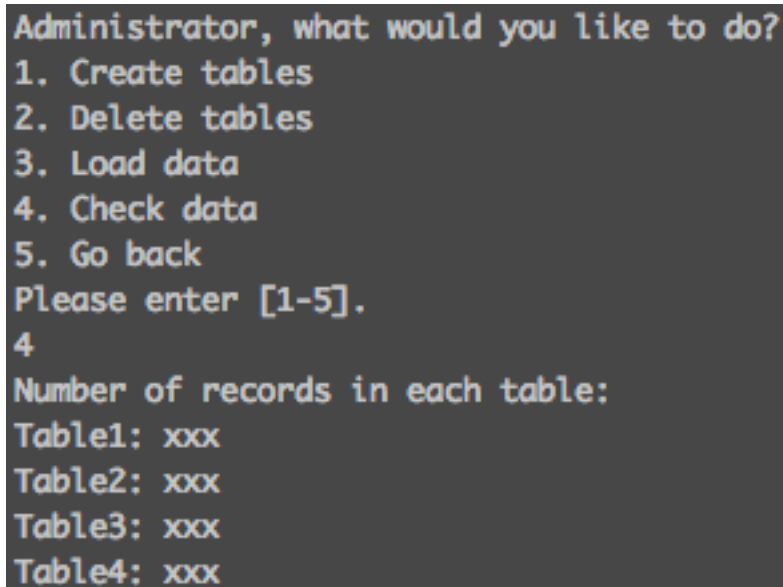


```
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
3
Please enter the folder path.
test_data
Processing...Data is loaded!
```

Figure 3: Example input and output of loading data.

- **Check data**

An administrator can check the data in the database. The system should display the number of records in each table in the database.



```
Administrator, what would you like to do?
1. Create tables
2. Delete tables
3. Load data
4. Check data
5. Go back
Please enter [1-5].
4
Number of records in each table:
Table1: xxx
Table2: xxx
Table3: xxx
Table4: xxx
```

Figure 4: Example output of showing number of records in each table.

5.2. Passenger

The system should let passengers to perform the following operations:

- **Request a ride**

A passenger can place requests with the criteria:

1. Number of passengers
2. The earliest model year of the vehicle (optional)
3. The partial model of the vehicle (optional)

The system should let the passenger enter his/her ID, and the criteria. He/she can press enter without any input to skip the optional criteria. The system should then search for any vehicles which are available (not in unfinished trips) and meets all the criteria: for criterion 1, all vehicles returned must have a number of seats which is greater than or equal to the request number of passengers; for criterion 2, all vehicles returned must have a model year which is later than or equal to the request model year; for criterion 3, all vehicles returned must have a model containing the keyword.

The partial matching should be case-insensitive and the input from the passenger is treated as a whole as illustrated in the followings. If the passenger enters “toyota”, the system may return a vehicle with the model “Toyota Prius”. If the passenger enters “toyota corolla”, then the system should not split the input, and should not return “Toyota Prius”.

If there are any vehicles which meet all the criteria, then the system should create a new request and display a success message including the possible number of drivers who will take the request, including drivers in unfinished trips. Otherwise display a message telling the passenger to adjust the criteria.

```

Passenger, what would you like to do?
1. Request a ride
2. Check trip records
3. Rate a trip
4. Go back
Please enter [1-4].
1
Please enter your ID.
1
Please enter the number of passengers.
4
Please enter the earliest model year. (Press enter to skip)
2015
Please enter the model. (Press enter to skip)
toyota
Your request is placed. 14 drivers are able to take the request.
  
```

Figure 5: Example input and output of requesting a ride.

- **Check trip records**

A passenger can check all his/her finished trips within a certain period. The system should let the passenger enter his/her ID, the start date, and the end date (for date format see section 4.1), then display each matching trip by its ID, the driver's name, the vehicle ID, the vehicles' model, the start time, the end time, the fee, and the rating, sorted in descending order of the start time.

```

Passenger, what would you like to do?
1. Request a ride
2. Check trip records
3. Rate a trip
4. Go back
Please enter [1-4].
2
Please enter your ID.
1
Please enter the start date.
2018-09-01
Please enter the end date.
2018-10-31
Trip ID, Driver Name, Vehicle ID, Vehicle model, Start, End, Fee, Rating
500, Albert Chung, BP9474, Nissan Serena, 2018-10-30 18:01:02, 2018-10-30 19:05:
54, 64, 1
485, Denise Kwok, RC3611, Mercedes-Benz C 200, 2018-10-23 05:27:22, 2018-10-23 0
6:22:14, 54, 0
437, Stuart Fung, LI7059, Honda Freed G Aero, 2018-09-19 11:51:40, 2018-09-19 13
:14:12, 82, 2
428, Ada Au-Yeung, VD7673, Toyota Echo, 2018-09-14 03:26:08, 2018-09-14 04:28:24
, 62, 5

```

Figure 6: Example input and output of checking trip records.

- **Rate a trip**

A passenger can rate his/her finished trips by giving a rating between 1 to 5. The system should let the passenger enter his/her ID, the trip's ID, and a rating, then look for the trip by the trip ID, set the rating, and display the trip by its ID, the driver's name, the vehicle ID, the vehicles' model, the start time, the end time, the fee, and the rating.

```
Passenger, what would you like to do?
1. Request a ride
2. Check trip records
3. Rate a trip
4. Go back
Please enter [1-4].
3
Please enter your ID.
1
Please enter the trip ID.
485
Please enter the rating.
5
Trip ID, Driver name, Vehicle ID, Vehicle Model, Start, End, Fee, Rating
485, Denise Kwok, RC3611, Mercedes-Benz C 200, 2018-10-23 05:27:22, 2018-10-23 0
6:22:14, 54, 5
```

Figure 7: Example input and output of rating last trip.

5.3. Driver

▪ Take a request

A driver who is not in an unfinished trip can take a request as long as his/her vehicle satisfies all the criteria of a request:

1. The vehicle should have enough seats;
2. The vehicle should be new enough as specified by the earliest model year;
3. The model of the vehicle should match the request.

The system should let the driver enter his/her ID, then display each open request for which the driver is qualified by its ID, the passenger's name, and the number of passengers. The system should let the driver pick a request by entering a request ID, then mark the request taken, create a new trip, and display the trip by its ID, the passenger's name, and the start time.

```
Driver, what would you like to do?
1. Take a request
2. Finish a trip
3. Check driver rating
4. Go back
Please enter [1-4].
1
Please enter your ID.
20
Request ID, Passenger name, Passengers
1, Audrey Fu, 4
Please enter the request ID.
1
Trip ID, Passenger name, Start
501, Audrey Fu, 2018-09-03 22:32:01
```

Figure 8: Example input and output of taking request.

- **Finish a trip**

A driver who is in an unfinished trip can finish his/her current trip. The system should let the driver enter his/her ID, then look for an unfinished trip by this driver, display the trip by its ID, the passenger's id, and the start time. Then the system should let the driver confirm if he/she wants to finish the trip. If so, record the end time and calculate the fee for the trip. The fee should be the duration of the trip in minutes, rounded down to the nearest integer. The system should display the trip by its ID, the passenger's name, the start time, the end time and the fee.

```
Driver, what would you like to do?
1. Take a request
2. Finish a trip
3. Check driver rating
4. Go back
Please enter [1-4].
2
Please enter your ID.
20
Trip ID, Passenger ID, Start
501, 1, 2018-09-03 22:32:01
Do you wish to finish the trip? [y/n]
y
Trip ID, Passenger name, Start, End, Fee
501, Audrey Fu, 2018-09-03 22:32:01, 2018-09-03 22:41:55, 9
```

Figure 9: Example input and output of finishing trip.

- **Check driver rating**

A driver can check his/her driver rating. A driver's driver rating is defined by the average of the ratings of his/her last 5 rated. If a driver has less than 5 rated trips, then his/her driver rating is not yet determined until he/she finishes more trips get asks the passengers to rate them. The system should let the driver enter his/he ID. then display the driver rating, rounded to 2 decimal places, if it is determined. Otherwise, display a message telling the driver that his/her driver rating is not yet determined.

```
Driver, what would you like to do?
1. Take a request
2. Finish a trip
3. Check driver rating
4. Go back
Please enter [1-4].
3
Please enter your ID.
15
Your driver rating is 4.4.
```

Figure 10: Example input and output of checking driver rating.

5.4. Error Handling

If there is a runtime error, including data not found, incorrect format, incorrect content, and etc. The system should display an error message in human readable form as the example below.

```
Welcome! Who are you?
1. An administrator
2. A passenger
3. A driver
4. None of the above
Please enter [1-4].
5
[ERROR] Invalid input.
Please enter [1-4].
2
Passenger, what would you like to do?
1. Request a ride
2. Check trip records
3. Rate a trip
4. Go back
Please enter [1-4].
1
Please enter your ID.
501
Please enter the number of passengers.
3
Please enter the earliest model year. (Press enter to skip)

Please enter the model. (Press enter to skip)

[ERROR] Passenger not found.
```

Figure 11: Example output when errors occur.

6. Grading Policy

The marks are distributed as follows:

Phase	Content	Mark Distribution
1	ER-diagram	10%
	Relational schema (based on your ER-diagram)	10%
2	Java application	80%

- There will be a mark deduction if your application is terminated unexpectedly during the demonstration.
- You are not allowed to modify any source code during the demonstration.
- Every member in the same group will receive the same marks for the project. In order to encourage every student to participate in the project, a question about this project may be asked in the final examination.

7. Demonstration

- Every group has to sign up for a demonstration of its system. A registration form will be posted in Blackboard later.
- Every group member should attend the demonstration.
- Each demonstration will last for about 30 minutes.
- We will **compile** and **test** your system on a Linux 64-bit machine of the CSE department.
- The data files used in the demonstration may be different from the data files provided for testing.

8. Submission Methods

8.1. Phase 1

- Submit a PDF file (one copy for each group) to the collection box in Blackboard.
- The PDF file should include of an ER diagram, a relational schema, the group number, the names and the student IDs of all group members of your group.

8.2. Phase 2

- Submit a ZIP file (one copy for each group) to the collection box in Blackboard. The ZIP file should include all your source codes and a README file, which includes:
 - Your group number
 - The name and the student ID of each group member
 - Instructions on how to compile and run your system