Computer Programming and Database Systems

Academic Year 2020-2021 Term II

Course Project Assignment

Specification release date 28.03.2021

Submission deadline: 02.04.2021 by 12:00 noon on Blackboard (automatic sharp closing time)

Description

The goal of the project is to build a simplified and proof-of-concept software application for managing an electronic table reservation book for a restaurant in the evening of a specific day. The software is composed of a user interface and business logic part written in Python and an SQL database management system to store the data.

Application domain model. We suppose the restaurant has a certain number of tables, numbered sequentially with integers from 1 to n ($n \geq 9$). The restaurant manager, which is our software user, can receive table reservation requests and operate on the software in order to update the booking records in the system. A reservation is requested by a guest that tells the manager

- his/her name
- personal phone number
- number of guests for the table

The date and time are unnecessary information since, for sake of simplicity, we are supposing the software manages only one evening. The reservation is accepted if there is a table with the right capacity: among the tables that can accommodate the guests, one with the least number of seats gets actually reserved. The guests are not allowed to book with a name or phone number already existing in the reservations. The guests can request the cancellation of a reservation by providing name or phone number.

User interaction. The restaurant manager can interact with the system with *regular* operations that are intended to manage guest reservations:

- Record the reservation of a table
- Show a reservation
- Cancel a reservation
- List all the reservations
- List all the unreserved tables

Moreover, the software can fulfill the following simple *statistical* operations:

- Show the number of reserved tables
- Show the number of tables reserved with a specified number of guests
- Show the number of booked guests overall
- Show the number of unreserved seats overall
- Show the table(s) with the greatest number of unreserved seats
- Show the reserved table(s) with the greatest number of unreserved seats

The program shall show an error (as specified later on) in case any of the operations above cannot be accomplished (e.g. adding an existing reservation, adding a reservation that cannot be accomodated, showing or cancelling a non-existing reservation, …).

Program specification. The Python program, that communicates wih the DBMS storing the data, shall repeatedly: 1) output only the prompt character > ("greater than" symbol) in a line, and 2) wait for a user's *command* as input, and then 3) possibly output the corresponding outcome, according to the following specifications:

Command Syntax	**Description**

`R` *g phoneNumber bookingName*	Reserve a table for *g* guest(s) under the name *bookingName* with number *phoneNumber*, choosing one least capacity table as specified above. **Assumptions** (for sake of simplicity): *phoneNumber* is a string of decimal digits, *bookingName* is a single word formed by letters only.
	Example commands:
	`R 3 35452489 Martin`
	`R 5 0214327 Anna`

`S` *phoneNumberOrBookingName*	Show the information of a certain reservation identified by *phoneNumberOrBookingName* (that must be a phone number or a booking name), with output in the format: *t g s phoneNumber bookingName*, where *t* is the table number, *g* is the number of booked guests and *s* is the total number of seats for the table *t*. Example output: `7 3 5 35452489 Martin`
`C` *phoneNumberOrBookingName*	Cancel a reservation identified by *phoneNumberOrBookingName* Example command: `C Anna`

`L`	List all the reservations, one per line complying with the same format as for the `S` command's output.
	Example output:
	`7 3 5 35452489 Martin`
	`9 5 5 0214327 Anna`

`U`	List all the unreserved tables, one per line complying with the format: *t s*, where *t* is the table number and *s* is the number of seats for the table *t*.
`NT`	Output the number of reserved tables
`NT` *g*	Output the number of tables that have a reservation for *g* guests each.
`NG`	Output the number of booked guests overall
`NU`	Output the number of unreserved seats overall
`GU`	Show the information about table(s) with the greatest number of unreserved seats, one per line with the format: *t g s*, where *t* is the table number, *g* is the number of booked guests (0 if unreserved) and *s* is the total number of seats for the table *t*. Example output:

	2 7 9 7 3 5 8 0 2
GR	Show the information about reserved table(s) with the greatest number of unreserved seats, one per line. Output similar to GU command, but the field *g* cannot be 0.
X	Exit (close) the program

As written above, the tasks 1), 2) and 3) must be repeated until the user exits using the E command.

In some cases the operations R, S and C cannot be accomplished, e.g.: no table available for the reservation request, booking with the same name, showing or cancelling a non-existing reservation. In these cases the program shall output the string: Error

As for some other operations, the expected output might be empty: in such a case the program shall output the string: No result(s)

IMPORTANT: Strictly abide by the input-output specification of the commands above. Notice the single space character separating the parameter(s) in some commands or outputs.

The software architecture is simple: the user interacts with the Python program that communicates with a DBMS storing the data and running on the local machine.

Database. The SQL DBMS must contain at least two relations named: *dining_table* and *reservation*. For the relations you build, you have to choose the appropriate attributes with suitable names and data types, as well as the appropriate primary key(s) and foreign key(s). The database instance must be properly populated in the beginning with suitable dining table records at your choice (create at least some tables having 2, 3, 4, 5 and 6 seats). No reservations in the beginning. Choose the DBMS between: MySQL, PostgreSQL or SQLite.

Program usage example. Here is an example of software-user interaction with input/output:

```
>U
1 4
2 4
3 2
4 5
5 6
6 8
7 2
8 2
9 3
>L
No result(s)
>R 4 394753987 Jo
>R 7 83473924 Gary
>R 6 25345232 Jo
Error
>L
1 4 4 394753987 Jo
6 7 8 83473924 Gary
>C 7539870109
Error
>C 394753987
```

```
>R 3 394753987 Alon
>R 6 44166526 Sean
>L
5 6 6 44166526 Sean
6 7 8 83473924 Gary
9 3 3 394753987 Alon
>NT 6
1
>NG
16
>GR
6 7 8
>U
1 4
2 4
3 2
4 5
7 2
8 2
>X
```

Deliverables

You have to submit the following files through Blackboard's Assessments section:

1) The **source code** of the Python program which the user (restaurant manager) shall interact with. The code shall be properly commented, avoiding trivial comments but only providing information useful for clarifying difficult code parts. Necessary libraries shall be just properly imported. The code shall define a `main()` function that will be the entry point of the program as from user perspective.
2) A **SQL script** (a plain text file named with extension .sql) containing the DDL commands for creating the relations and the DML commands for populating the DBMS, in the correct order.
3) A **report document**, in Word or PDF format (maximum 2 pages), containing:
    A. a simple function diagram with boxes (representing functions) and arrows (representing possible calls) for your Python source code: one box for each function you defined, and one arrow from box (associated to function) A to box (associated to function) B if the code block of A contains an expression calling the function B
    B. the database schema written in the usual notation: use the same relation and attribute names used in the DDL, underline the primary key(s). Write a note explaining the foreign key(s)
    C. name and version number of the DBMS you used, the database connection username and password if necessary, name of your operating system
    D. your choice of the parameter n, and any other relevant implementation details that you consider as worth noting
4) A simple text file (name it with extension .txt) with an **example of input-output interactions** by the user with your program, similar to the usage example above. Be sure to include each kind of command in your example.

Your software must be correct from Python and SQL syntax viewpoint. Works raising language syntax errors will not be assessed. If any specification detail is missing in this document, make a reasonable and proper decision for the implementation on your own and explain it in the report document. Good luck and do your best!