Module: CMP4010B Database Systems
Assignment: SQL Coursework

Set by: Farhana Ferdousi Liza <F.Liza@uea.ac.uk>
Date set: 3rd March 2023
Value: 50%

Date due: 5th May 2023, 3:00 PM (for prototype with your own test data) and 9th May 2023, 07:00 PM (for completed assessment template with assessment data)
Returned by: 8 June 2023
Submission: Blackboard
Checked by: Professor Beatriz De La Iglesia <B.Iglesia@uea.ac.uk>

Learning outcomes

Experience in the following:

- problem solving techniques using SQL, PostgreSQL; (SQL, Transferable skills)

- interpreting user requirement and defining solutions; (SQL, DBMS understanding)

- creating table definitions using SQL; (SQL, Relational Model)

- manipulating table data using SQL; (SQL)

- SQL programming in PostgreSQL; (SQL)

- SQL transactions in PostgreSQL; (SQL, Transactions)

- demonstration and presentation of technical systems; (Transferable skills)

- managing time based on workload, deadlines, and distribution of effort. (Transferable skills)

Specification Overview

Aim:

To implement part of a database application by first completing the database table definitions, then writing interactive SQL statements to interact with the database.

Description

See Attached.

Relationship to formative assessment

The SQL lab in week 6 is a formative assessment for this. Students can mark their own effort for that lab exercise to understand how you may do in the coursework and what you need to improve on. Also, all SQL Labs build towards this exercise, with each lab helping you to perform tasks that contribute to building this solution. Labs include expected query results so you can ensure you get correct answers and perform a form of self-assessment.

Deliverables

Summary of deliverables:

Part 1: A copy of your SQL data definition statements and your own test data should be produced and submitted electronically by 5th May 2023, 3 PM.

Part 2: Your SQL data manipulation statements for each of the requirements (See **transactions of interest**) should be produced and submitted electronically together with evidence of testing each statement with your own test data by 5th of May 2023, 3 PM and with the assessment (test) data by 9th May 2023, 7 PM.

The Assessment Data and Assessment Template file will be issued nearer the submission time in Week 10.

Resources

The necessary materials are in the lecture notes/lab materials. Links to relevant PosgreSQL help pages are in the lecture slides. Recommended books in reading list are also useful.

Marking scheme

Assessment criteria

- Good use of SQL data definition language to complete the table definitions;
- Good use of SQL data manipulation language to write interactive queries;
- Ability to correctly and accurately interpret project specification. This may include a little bit of independent thinking on what may make a table/report look good;
- Correct functionality and output as required by each requirement;
- Neatly presented work with correct program output during demonstration; Sufficiency and completeness of Submitted code and tests data.
- Approximately 25% of marks will be allocated to the DDL in Part 1; Approximately 70% of marks will be allocated to the DML in Part 2; Approximately 5% of marks will be allocated to the quality of the submission;

Plagiarism, collusion, and contract cheating

The University takes academic integrity very seriously. You must not commit plagiarism, collusion, or contract cheating in your submitted work. Our Policy on Plagiarism, Collusion, and Contract Cheating explains:

- what is meant by the terms ‘plagiarism’, ‘collusion’, and ‘contract cheating’
- how to avoid plagiarism, collusion, and contract cheating
- using a proof reader
- what will happen if we suspect that you have breached the policy.

It is essential that you read this policy and you undertake (or refresh your memory of) our school’s training on this. You can find the policy and related guidance here: https://my.uea.ac.uk/departments/learning-and-teaching/students/academic-cycle/regulations-and-discipline/plagiarism-awareness

The policy allows us to make some rules specific to this assessment. Note that:

In this assessment, working with others is *not* permitted. All aspects of your submission, including but not limited to: research, design, development and writing, must be your own work according to your own understanding of topics. Please pay careful attention to the definitions of contract cheating, plagiarism and collusion in the policy and ask your module organiser if you are unsure about anything.

CMP-4010B Database Systems
SQL Coursework

Introduction

Your company has been given the fund to implement the system for the new domestic Cheap & Cheerful Airways (CCA) database system. Your role is to prototype and test the functionality required for the booking aspect of the system.

The first stage in this process is to analyse the requirements and write SQL statements to perform these tasks. These statements can be tested using an interactive SQL interface to ensure correct functionality.

A description of the tables and required functionality has been provided. Naturally, it is grossly simplified compared with a real system. A detailed specification of the task to be undertaken and the deliverables to be produced for assessment are given below.

You may, of course, use your own facilities to develop your program but the final version **must use PostgreSQL and run in the CMP labs.**

System Functionality

The database comprises the following tables:

LeadCustomer (CustomerID, FirstName, Surname, BillingAddress, email)

Passenger(PassengerID, FirstName, Surname, PassportNo, Nationality, DoB)

Flight (FlightID, FlightDate, Origin, Destination, MaxCapacity, PricePerSeat)

FlightBooking (BookingID, CustomerID, FlightID, NumSeats, Status, BookingTime, TotalCost)

SeatBooking(BookingID, PassengerID, SeatNumber)

where the Status is an attribute with one character length and can be one of the following:

- Reserved (R) – the seats have been allocated and the booking has been completed

- Cancelled (C) – the booking has been cancelled and seats are no longer allocated

Calculate the 'Available' status based on the current flight information including maximum capacity, reserved and cancelled seats.

Notes - The above tables are ***designed*** based on the following requirements where several **assumptions** have been made to reduce the volume of programming required:

- A flight is a simple one-way flight which describes the maximum capacity of the aircraft to which it has already been allocated. You do not need to worry about booking passengers a return trip.
- A lead customer can book seats for multiple passengers; the seats must be all on the same flight.
- The origin and destination are simple text fields and not using any form of lookup table.
- Each flight has a price per seat and all seats in a flight are charged at that price.
- The total cost for the booking is the sum of all individual seat costs.
- The lead customer may or may not be one of the passengers in the booked flight.
- The system uses default "ISO, DMY" date styles (SET datestyle='ISO, DMY')

The **transactions of interest** for CCA are as below:

A. Insert a new record. This could be
   a. Given a lead customer ID number, name, and contact details, create a new customer record.
   b. Given a passenger with an ID, name, date of birth, etc., create a new passenger record.
   c. Given a flight ID number, origin, destination, flight date, capacity of the aircraft, and price per seat create a new flight record.

B. Given a customer ID number, remove the record for that customer. It should not be possible to remove customers that have active (i.e., reserved) flight bookings. A customer that has only cancelled bookings could be removed; the associated bookings should also be removed along with all the seat bookings.

C. Check the availability of seats on all flights by showing the flight ID number, flight date along with the number of booked seats, number of available seats and maximum capacity.

D. Given a flight ID number, check the status of all seats currently allocated to that flight, i.e., return the total number of reserved/ cancelled/ available seats.

E. Produce a ranked list of all lead customers, showing their ID, their full name, the total number of bookings made, and the total spend made for all bookings. The list should be sorted by decreasing total value.

F. Given a booking ID, customer ID number, flight ID number, number of seats required and passenger details, make a booking for a given flight. This procedure should first show seats available in a given flight and then proceed to insert booking, if there are sufficient seats available. The customer could be an existing customer or a new customer, in which case it should be entered first into the database. Seats numbers can be allocated at the time of booking or later on. The making of a booking with all the steps outlined should work as an atomic operation.

G. Given a booking ID number, cancel the booking. Note that cancelling a booking only changes the status and should not delete the historical details of the original booking. However, cancelled seats should be viewed as available.

To be able to provide support to the above CCA intended transactions, you will have to use Data Definition Language to create the tables and add integrity constraints to have a consistent and reliable database for CCA. In Part 1, your task is to design and develop the data definition language incorporating integrity constraints.

Part 1: Database definition and loading (approx. 25% of marks)

Prepare an SQL program to create your copy of the CCA database. This should take, as a starting point, the table definitions given in the Appendix, but you also need to prepare additional SQL clauses and/or statements to complete the definition of the database by specifying primary keys, domain constraints, entity, and referential integrity constraints, etc. Note that, you should NOT modify the name and type of the attributes (i.e., the information you have been given in the Appendix: Minimal database definition). Save all your Data Definition Language (DDL) statements in a text file.

At this stage the tables are empty. Load (e.g., using insert statements) a reasonable volume of your chosen test data into the tables for prototyping. Your own test data should be sufficient to test the queries (i.e., transactions of interest) for the prototype phase with their expected output and should provide a suitable environment in which to test normal operation as well as abnormal conditions (e.g., if you add constraint on an attribute by defining a primary key, adding multiple rows with same value for that attribute would be considered as an abnormal condition).

To summarise:

- Document this stage in a text file with a copy of your complete SQL DDL statements (including any table definitions, views, triggers, comments, etc).
- Produce a copy of the chosen test data you loaded for testing the prototype and submit as a text file.

You will submit Part 1 deliverables by 5th May 2023 at 3 pm.

Part 2. Interactive SQL version of the transactions (approx. 70% of marks)

Prepare and test interactive SQL statements for the transactions of interest. Test these statements using the SQL editor in PostgreSQL. You may need more than one

execution of some of the task types to verify the correctness of the transaction of interest (e.g., test primary keys, referential integrity, correct and incorrect execution, etc.). Please be prepared to add the SQL statements for a submission template which will be released to you on week 10 along with the assessment test data. The template will look like the exemplar template in the Appendix (see section Template for Submission with a Mock Question and Answer).

To summarise:

- **Document the SQL statements and Evidence of testing with your own test dataset**: Document this stage with a copy of SQL statements. To elaborate this stage, read each transaction of interest (A … G), write SQL statements to execute transactions, and document them in a document following the formatting of Appendix template (i.e., Template for Submission with a Mock Question and Answer). For each SQL statement, please document the output of running the query at least once but you should perform further testing to ensure the correctness of the SQL statements. Please pay attention to the quality of testing with your chosen data. You will submit the prototype with the own test data data by 5th May 2023 at 3 pm.

- **Executing SQL statements on assessment (test) dataset and preparing the submission template**: On 5th May 2023 at 5 pm, you will get access to the assessment test data and template for submission (assessment template). You will execute the query on your developed SQL statements at the prototype phase which is submitted by the 5th May 2023, 3pm using the assessment dataset and complete the submission template. You will need to submit the submission template by 9th May 2023 at 7 pm.

Note: Please name the prototype submission folder and assessment template with your student id. The SQL statements with your chosen own test datadata and assessment (test) data ***must match***.

Quality of Deliverables (approx. 5% of marks):

- Electronic submission to Blackboard (All files and subfolders to be collected in a folder named with your student number, zipped and submitted following instructions in the Blackboard):
    - Part 1: A copy of your design including SQL data definition statements, functions, triggers together with your own test data.
    - Part 2: Your SQL data manipulation statements (annotated with comments if necessary) for each of the **Transactions of interest** plus evidence of testing (with your chosen own test data by 5th May 2023, 3 PM and assessment (test) data along with completed assessment template by 9th May 2023, 7 PM).

If you submit the prototype by the 5th May 2023 deadline, and don't submit the assessment template with the assessment data (or vice versa), you will get the zero score for the coursework. You need to submit both the prototype version of your work by 5th May 2023, 3 PM and assessment template with assessment data by 9th May 2023, 7 PM. The marks will be allocated based on the quality of

the prototype submission, and correctness of the prototype based on the assessment data in assessment template.

Important notes

- The document you submit should be complete and neatly formatted to ease reading.
- This is an individual piece of coursework NOT a group project. Collusion/plagiarism checks may be carried out. If prototype and assessment version of the SQL statement does not match, the system will flag for further Collusion/Plagiarism checks.
- As you will be given the assessment data based on the tables in minimal database definition in week 10 to load assessment (test) data into your tables, it is vital that you do not change the table names, field names, field types etc. from the description in Appendix (see section, Minimal database definition).

Appendix

Minimal database definition

The following SQL program provides a minimal definition of the CCA database for use in the Part 1. A copy of this text can be found in the file CgTableDef.txt in Blackboard. To do Part 1 of the assignment you will probably want to *add* SQL clauses or statements.

```
CREATE TABLE LeadCustomer
(
        CustomerID INTEGER NOT NULL,
        FirstName VARCHAR(20)  NOT NULL,
        Surname VARCHAR(40)  NOT NULL,
        BillingAddress VARCHAR(200)  NOT NULL,
        email  VARCHAR(30) NOT NULL
)
CREATE TABLE Passenger
(
        PassengerID INTEGER NOT NULL,
        FirstName VARCHAR(20)  NOT NULL,
        Surname VARCHAR(40)  NOT NULL,
        PassportNo VARCHAR(30) NOT NULL,
        Nationality VARCHAR(30) NOT NULL,
        Dob DATE NOT NULL
)
CREATE TABLE Flight
(
        FlightID INTEGER NOT NULL,
        FlightDate TIMESTAMP NOT NULL,
        Origin VARCHAR(30) NOT NULL,
        Destination VARCHAR(30) NOT NULL,
        MaxCapacity INTEGER NOT NULL,
        PricePerSeat DECIMAL NOT NULL

)
CREATE TABLE FlightBooking
(
        BookingID INTEGER NOT NULL,
        CustomerID INTEGER NOT NULL,
        FlightID INTEGER NOT NULL,
        NumSeats INTEGER NOT NULL,
        Status CHAR(1) NOT NULL,
        BookingTime TIMESTAMP NOT NULL,
        TotalCost DECIMAL

)
CREATE TABLE SeatBooking
(
        BookingID INTEGER NOT NULL,
        PassengerID INTEGER NOT NULL,
        SeatNumber CHAR(4) NOT NULL

)
```

Template For Submission with a Mock Question and Answer:

EXAMPLE OF HOW TO FILL THE SUBMISSION DOCUMENT (using indicative tasks) WITH YOUR OWN TEST DATA.

Testing task 1

1. Produce a query that counts the number of tuples in each table.

INSERT YOUR SQL QUERY AND OUTPUT HERE