School of Computing Sciences

Module: CMP-4010B Database Systems
Assignment: SQL and Node.js Programming Exercise

Set by: Dr Beatriz de la Iglesia, <b.iglesia@uea.ac.uk>
Date set: 10/02/2020
Value: 40%

Date due: Wednesday, 29th April 2020, 3 pm (week 12)
Returned by: Assessment Period Week 3
Submission: Blackboard

Learning outcomes

Experience in the following:
- problem solving techniques using SQL, PostgreSQL and Node.js;
- interpreting user requirement and defining solutions;
- creating table definitions using SQL;
- manipulating table data using SQL;
- SQL programming in postgreSQL;
- SQL transactions in PostgreSQL;
- managing a database from a Node.js application
- demonstration and presentation of technical systems;
- managing time based on workload, deadlines and distribution of effort.

Specification

Overview

Aim:
To implement a database application in Node.js by first completing the database table definitions and writing interactive SQL statements.

Description

See attached.

Relationship to formative assessment

We do not have explicit formative assessment. However, SQL Labs build towards the 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.

Resources

The necessary materials are in the lecture notes/seminar exercises/lab materials, as well as additional reading folder. Links to relevant PosgreSQL help pages are in reading folder. Recommended books in reading list can also help.

Deliverables

Summary of deliverables:

- Part 1: A copy of your SQL data definition statements and test data should be produced during the demonstration and submitted electronically.
- Part 2: Your SQL data manipulation statements for each of the requirements should be available during the demonstration and submitted electronically together with **evidence of testing.**
- Part 3: Your Node.js source code files ready to be executed in a folder called “4010B-node-<YOUR-STUDENT-ID>” .
- Attend and participate in a demonstration in week 13/14.

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;
Good use of Node.js to implement a prototype database client application;
Ability to correctly and accurately interpret project specification;
Correct functionality and output as required by each requirement;
Neatly presented work with correct program output during demonstration;
Submitted code and tests to support demonstration.

Approximately 25% of marks will be allocated to the DDL in Part 1; Approximately 48% of marks will be allocated to the DML in Part 2; Approximately 27% of marks will be allocated to the development of the Node.js application.

Handing in procedure

You will be expected to demonstrate your work in week 13/14 using the CMP lab computers (not your own computer). The SQL and program demonstrated must match the version handed in! All deliverables will be submitted electronically (instructions on how to submit electronically will be issued in due course). If deliverables are incomplete or do not match demonstrated code marks may be deducted accordingly.

Plagiarism

Plagiarism is the copying or close paraphrasing of published or unpublished work, including the work of another student without the use of quotation marks and due acknowledgement. Plagiarism is regarded a serious offence by the University and all cases will be reported to the Board of Examiners. Work that contains even small fragments of plagiarised material will be penalised.

CMP-4010B Database Systems

Coursework Assignment – SQL and JavaScript/Node.js Programming

Set: Week 5
Hand in: Wednesday, 29th April 2020 (Week 12) by 3 PM (Demonstrations will be scheduled for Week 12/13)
Returned by: Assessment period week 3
Value: 40% of unit marks

Introduction

Your company has been given the opportunity to implement the system for the new domestic Cheap & Chearful Airways (CCA) database system. Your role is to prototype and test some of 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 should be tested using an interactive SQL interface to ensure correct functionality before writing the programmatic interface. Once the correct functionality has been achieved, you are required to develop a Node.js application to execute SQL statements. The Node.js application should execute the commands developed earlier using the interactive SQL interface. Your prototype Node.js application will be used to demonstrate the processes and database functionality to the company before a full user interface is developed; therefore you only need to implement the features required for the user interaction described in the tests below. If you do not develop a Node.js application it will be possible to demonstrate your interactive SQL statements.

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.

System Functionality

The database comprises the following tables:

LeadCustomer (<u>CustomerID</u>, FirstName, Surname, BillingAddress, email)
Passenger(<u>PassengerID</u>, FirstName, Surname, PassportNo, Nationality, DoB)
Flight (<u>FlightID</u>, FlightDate, Origin, Destination, MaxCapacity, PricePerSeat)
FlightBooking (<u>BookingID</u>, CustomerID, FlightID, NumSeats, Status, BookingTime, TotalCost)
SeatBooking(<u>BookingID, PassengerID</u>, SeatNumber)

where the Status can be one of the following:

- Reserved – the seats have been reserved and the booking has been completed

- Cancelled – the booking has been cancelled and seats are no longer reserved

A number of assumptions have been made to reduce the volume of programming required, these are:

- A flight is a simple one-way flight which also 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. All passenger details are entered later, as part of a request for advanced passenger information when seats are allocated.
- 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 actions of interest to us in this work are:

1. 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 flight ID number, origin, destination, flight date, capacity of the aircraft, and price per seat create a new flight record.
    c. Given a passenger with an ID, name date of birth, etc., create a new passenger record.

    (Approximately 9% of marks)
2. 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.

    (Approximately 6% of marks)
3. Check the availability of seats on flights by showing the flight ID number, flight date along with the number of booked seats, number of available seats and maximum capacity. As a minimum you should be able check availability for all flights. Better solutions will be able to seach for a specific flight by flightID; by destination; or by date. Note that seats are booked (not available) as soon as the flight booking is entered and regardless of whether they have been associated with a particular passenger.

    (Approximately 13% of marks)
4. 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.

    (Approximately 7% of marks)

5. 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.

   (Approximately 8% of marks)

6. Create a new flight booking. This procedure can be broken down into a number of steps as follows:

   a. Provide a facility to search for existing customers (as a minimum by customerid but better solutions may provide other search options, e.g. by surname). If lead customer exists, we can use existing record. If new, we can enter in the database using query developed for 1a.

   b. Enter flight booking as requested, including total cost for the booking. The entering of a new booking should work as an atomic operation so either the whole booking (insert of lead customer if required, insert of flight booking itself) succeeds or it fails. Hence lead customers should not be added if the flight booking does not get added. If any problems occur during the booking (e.g. the seats are not available or any of the insert fails) the booking should be cancelled and all of the operations undone.

   (Approximately 18% of marks)

7. Allocate seats to each passenger associated with a booking. Passengers need to be entered using the query developed in 1 c. There should be no more seat bookings that those associated with the flight booking, e.g. if the flight booking was for 3 passengers only 3 seats should be allocated to 3 passengers.

   (Approximately 5% of marks)

8. 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.

   (Approximately 6% of marks)

Assignment

Part 1. Create a test database (25% of marks)

A copy of the create table statements for the database definition is available in a text file CWTables.txt in Blackboard. 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 particular you should not modify the type of the ID attributes to anything other than integer (i.e. do not use auto increment fields). Save all your Data Definition Language (DDL) statements in a text file.

Load a reasonable volume of test data (you should create this) into the tables for use in your testing. The test data should be sufficient to test all of the queries with their expected output and should provide a suitable environment in which to test normal operation as well as abnormal conditions.

- Document this stage with a copy of your complete DDL statements in SQL (including any table definitions, constraints, views, triggers, indexes, etc.) Also include a copy of your own test data used to test your queries. This could be in the form of insert statements or it could be a copy of the tables showing the inserted data. **You should bring a printed copy to the demonstration.** This must also be submitted electronically.

Part 2. Test the functionality (48% marks approx.)

For each of the tasks described above, prepare in text files interactive SQL statements. Test these statements using the PGAdmin 4 interface. You should ensure you test your queries thoroughly, e.g. test entity integrity, referential integrity and other constraints as necessary.

The purpose of this stage is to prototype and test SQL statements for each task for use in your final version of the Node.js application. However, your .sql files need to ready to be loaded during the demo so if you fail to demonstrate your Node.js program working you can at least demonstrate your prototype SQL statements through the interactive interface.

- Document this stage with a copy of the SQL Data Manipulation Language (DML) statements. **Your SQL statements need to be accessible (e.g. through copy/paste) as text files through the demonstration so they can be run in PGAdmin4 if necessary.**
- Document also with evidence of testing of each SQL statement (e.g. copy of the output of running the query).

Part 3. Develop a prototype version of the client (27% marks aprox.)

Write a Node.js application. The application should comprise a home web page with a number of links and forms each of which handles one of the tasks mentioned above. Each link or form’s action should lead to a Node.js function to execute the relevant SQL query. If the query produces an output this should be presented on a web page which should include a link back to the home page. If an error occurs a suitable error message should be displayed on the relevant web page. When a task is complete, a simple one line message should be displayed on the webpage to indicate the status of completion. Node.js must not be used for any data processing other than submitting SQL statements and receiving results.The objective is to demonstrate the good use of Node.js and SQL for database access. Whilst your program should be well laid out and easily readable, no extra credit will be given for complex coding and exotic user interfaces are not required!

- The deliverable for this part of the assignment will be a copy of the source code of your Node.js application to be submitted to Blackboard. Your source code should be placed in a folder called “4010B-node-<YOUR-STUDENT-ID>”. It is your responsibilty to ensure that the necessary code is in this folder so that an examiner can run your Node.js application from it. You **do not** need to produce a printed copy of your Node.js code for the demonstration.

All deliverables from part 1, 2 and 3 should be collated together in a folder which should then be zipped ready for electronic submission by the **coursework deadline**. Archive formats other than zip are **not acceptable**. Instruction for electronic submission of code will be accessible via Blackboard. Online submissions to Blackboard are timed by the system. Any submissions that are late will receive the standard penalties. Note that incomplete submissions for any of the parts will result in a reduced mark to account for the lack of supporting evidence for your work.

Week 12/13 Demonstration of Part 3

You must *keep* your tables, test data, interactive SQL statements and program so that you can give a demonstration to show your Node.js application or SQL statements working. Demonstrations will be scheduled for week 13/14 and **all marks will be awarded at the demonstration**. However, demo marks will be penalised and may even go down to zero if the supportive evidence is not submitted online as requested, is incomplete or if it is submitted late. You will only receive marks for what is submitted. In the case of SQL statements for part 2, you need to submit the statement as well as evidence that you tested it at least once (by showing the output of execution from the pgAdmin 4 interface). Demonstrating your work (along with submitting the requested evidence) is a *mandatory* part of the assessment procedure and those that do not come for their scheduled demonstration time will receive a mark of zero for the assignment, unless they have secured an extension. The demonstration will involve loading a set of *provided* test data and carrying out a standard set of tests. The SQL and program demonstrated must match the version submitted electronically! No changes to the code are allowed after the submission deadline.

Important Notes

- No additional report is required.
- The electronic documents should be well presented to aid checking.
- This is an individual piece of coursework **NOT** a group project. Collusion checks may be carried out on the electronic submissions.
- As you will be given a SQL script in week 12 to load test data into your tables, it is vital that you do not change the table names, field names, field types etc. from what is described below.
- The demonstration must take place on a **CMP lab machine using a Node.js desktop application as the client communicating with the CMP PostgreSQL database server.** We will not accept other systems, languages, technologies or machines.

Summary of Deliverables

- Part 1: A copy of your SQL data definition statements and test data should be produced during the demonstration and submitted electronically.
- Part 2: Your SQL data manipulation statements for each of the requirements should be available during the demonstration and submitted electronically together with **evidence of testing.**
- Part 3: Your Node.js source code files ready to be executed in a folder called “4010B-node-<YOUR-STUDENT-ID>” .
- Attend and participate in a demonstration in week 13/14.

Appendix

Initial database design

```
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

)
```