SC-T-202-GAG1
Project 3 – PL/pgSQL Programming

1 Preparation

Readings: Ramakrishnan & Gehrke: Chapters 3 (esp. 3.6) and 5 (esp. 5.8-5.9).

Presentations and slides: Module 5: Coding with SQL (on Canvas).

2 Description

The project involves implementing functionality for a ticket-selling website, similar to www.tix.is and www.midi.is, using pl/pgsql programming code inside PostgreSQL.

A database has already been created, but due to security issues and the fact that more than one user interfaces use the database (website, app and local app for staff members), a demand has been issued for the data logic to be moved down to the database.

As part of the security precautions taken, these user interfaces will be connecting as a database user who will not have permission to change the underlying tables directly, but only access objects such as the ones created in this project. However, you do not have to worry about user access in this project, only the programming.

2.1 Database schema

The database consists of six tables:

- People (*ssn*, name, email, address, zipcode, userType)
- Venues (*id*, name, address, zipcode, number_of_seats)
- EventTypes (*id*, name)
- Events (*id*, name, created, event_description, event_listPrice, event_type)
- EventSchedules (*id*, event_id, venue_id, event_time, event_price, number_of_bookedSeats)
- Bookings (*schedule_id*, *seat_id*, people_ssn)

Note: Events have schedules, so each event like a specific movie, can be shown multiple times, on multiple venues. This is done with the EventSchedules table. Each person in the databaes can book one or more seat at any schedule by entering the Bookings table.

2.2 Database Installation Files

The given CREATe.sql script creates the table schema. There are also comments on every column.

The FILL.sql script will provide you with some randomly generated people, a few events and eventTypes, along with a few schedules so you can better understand how the database is designed.

3 Tasks

Your job is to create functions, triggers and views to complete the following tasks without changing the table schema:

1. Create a trigger on the Bookings table that throws a descriptive error if the given seat number does not exist in the corresponding venue.
2. Create a trigger on the EventSchedules table that makes sure that (i) each venue can only be booked once a day, and (ii) each event can only be scheduled once a day. If those rules are violated the trigger should throw a descriptive error and cancel the insertion.
3. Create a function fGetNextSeatAvailable that takes a schedule ID as input parameter and returns the next available seat number on that event.
4. Create a function fGetNumberOfFreeSeats that takes a schedule ID as input parameter and returns the current number of free seats on that scheduled event.
5. Create a trigger on the Bookings table that maintains the number_of_bookedSeats counter in the EventSchedules table. This counter shows how many seats have been booked at that given event at each time.
6. Create a function fFindConsecutiveSeats that takes as input parameters a scheduled event ID and the number of consecutive seats it should find. The function returns the first (lowest) seat number where there are sufficiently many free seats in a row. If no sequence of sufficiently many adjacent free seats exists, the function should throw a descriptive error.

   Note: As an example, consider the seat row below, where seats 3 and 7 are already booked. If asked to find a row of 1 or 2 seats, the output would be 1. If asked to find 3 seats, the output would be 4. If asked to find 4+ seats, an exception would be thrown.

1	2	3	4	5	6	7	8	9	10

7. Create a procedure, fBookManySeats that takes 4 input parameters (schedule ID, customer ssn, first seat number and how many seats it should book) and makes a booking for the given number of seats, starting on the first seat number provided and booking consecutive seats to the given customer.
8. Create a procedure, fFindAndBookSeats that takes number of consecutive seats that should be booked, the schedule ID and customer ssn. This procedure books the next available seat row with given amount of seats in a row to the given schedule to the given person. If seat row is not found, a descriptive error should be thrown describing the problem.

   Note: Use the functions created above, fFindConsecutiveSeats and fBookManySeats.

9. Create a view vGetShowList that returns a list of schedules that have not already passed. The list should show ID of schedule, ID of event, time of event, name of event, name of venue, total number of seats and how many seats are still available.
10. Create a view vListOfVipPeople that shows ssn, name and email of all people that have booked every event that has been scheduled in the current year.

    Note: While an event can be scheduled many times in a year, and also in more than one year, it is not necessary to be booked in every instance of the event in the current year. Attending any one of the scheduled instances is sufficient.

4 Additional Hints

- Check out the function generate_series() in PostgreSQL documentation. This method could help you with some of the tasks.
- Functions that you create can be used in other functions/views.
- You will have to provide yourself with some test data as you go on, in addition to the data present in the FILL.sql script.
- You will need to test your code thoroughly; you can start that work with Exercise 06.

5 Groups

The project is a group project. Each group must have 2-3 students, and students must form groups themselves.

Note: It is possible to get an exemption and submit alone, if for some reason collaboration is *completely impossible*. In order to request an exemption, please send mail to gylfig@ru.is. No exemptions will be made for larger groups!

6 Deliverable

The solution takes the form on one text file, called **SOL.sql**, that contains all the SQL code required to implement the functionality described above. The SQL code must be easily understandable (well written, consistently formatted, and well documented).

Solutions must be submitted on Canvas.

Late submissions will not be accepted, so make sure to submit your solutions on time.