

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Полухин А.В.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание (Вариант 19. БД «Пассажир»)	3
Описание предметной области:	3
Задание 4. Создать хранимые процедуры:	3
Задание 5. Создать необходимые триггеры	3
Выполнение	3
Создание хранимых процедур:	4
1. Для повышения цен в пригородные поезда на 20%.	4
2. Для создания нового рейса на поезд.	5
3. Для формирования общей выручки по продаже билетов за сутки.	7
Создание необходимых триггеров:	8
Вывод	10

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание (Вариант 19. БД «Пассажир»)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Описание предметной области:

Информационная система служит для продажи железнодорожных билетов. Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5%. Билет может быть приобретен в кассе или онлайн. Если билет приобретен в кассе, необходимо знать, в какой. Для каждой кассы известны номер и адрес. Кассы могут располагаться в различных населенных пунктах.

Поезда курсируют по расписанию, но могут назначаться дополнительные поезда на заданный период или определенные даты.

По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные пункты по маршруту.

БД должна содержать следующий минимальный набор сведений: Номер поезда. Название поезда. Тип поезда. Пункт назначения. Пункт назначения для проданного билета. Номер вагона. Тип вагона. Количество мест в вагоне. Цена билета. Дата отправления. Дата прибытия. Дата прибытия для пункта назначения проданного билета. Время отправления. Номер вагона в поезде. Номер билета. Место. Тип места. Фамилия пассажира. Имя пассажира. Отчество пассажира. Паспортные данные.

Задание 4. Создать хранимые процедуры:

1. Для повышения цен в пригородные поезда на 20%.
2. Для создания нового рейса на поезд.
3. Для формирования общей выручки по продаже билетов за сутки.

Задание 5. Создать необходимые триггеры

Выполнение

Создание хранимых процедур:

1. Для повышения цен в пригородные поезда на 20%.

```
Passenger=# select Passenger.up_price_to_suburban_trains();
up_price_to_suburban_trains
-----
```

(1 строка)

```
Passenger=# create or replace function passenger.up_price_to_sub
urban_trains(
Passenger(# ) returns void as $up_price_to_suburban_trains$
Passenger$$ begin
Passenger$$     UPDATE passenger.seats
Passenger$$     SET price = price * 1.2
Passenger$$     WHERE several_carriages_id in
Passenger$$     (
Passenger$$         select
Passenger$$             several_carriages.several_carria
ges_id
Passenger$$         from
Passenger$$             passenger.trains,
Passenger$$             passenger.timetable,
Passenger$$             passenger.several_carriages,
Passenger$$             passenger.seats
Passenger$$         where
Passenger$$             train_type = 'suburban'
Passenger$$             and several_carriages.several_ca
rriages_id = seats.several_carriages_id
Passenger$$             and seats.seat_status = true
Passenger$$             and trains.train_id = timetable.
train_id
Passenger$$             and passenger.several_carriages.
timetable_id = passenger.timetable.timetable_id
Passenger$$     );
Passenger$$ end;
Passenger$$ $up_price_to_suburban_trains$ language plpgsql;
CREATE FUNCTION
```

Данные в базе данных до и после:

18		1a306,37	?		suburban
1		1a306,37	?		suburban
2		1a306,37	?		suburban
3		1a306,37	?		suburban
4		1a306,37	?		suburban
5		1a306,37	?		suburban
6		1a306,37	?		suburban
7		1a306,37	?		suburban
8		1a306,37	?		suburban
9		1a306,37	?		suburban
10		1a306,37	?		suburban
11		1a306,37	?		suburban
12		1a306,37	?		suburban
13		1a306,37	?		suburban
14		1a306,37	?		suburban
15		1a306,37	?		suburban
16		1a306,37	?		suburban
17		1a306,37	?		suburban
1		4a320,00	?		suburban
2		4a320,00	?		suburban
3		4a320,00	?		suburban
4		4a320,00	?		suburban
5		4a320,00	?		suburban
6		4a320,00	?		suburban
7		4a320,00	?		suburban
8		4a320,00	?		suburban
9		4a320,00	?		suburban

18		1a567,64	?		suburban
1		1a567,64	?		suburban
2		1a567,64	?		suburban
3		1a567,64	?		suburban
4		1a567,64	?		suburban
5		1a567,64	?		suburban
6		1a567,64	?		suburban
7		1a567,64	?		suburban
8		1a567,64	?		suburban
9		1a567,64	?		suburban
10		1a567,64	?		suburban
11		1a567,64	?		suburban
12		1a567,64	?		suburban
13		1a567,64	?		suburban
14		1a567,64	?		suburban
15		1a567,64	?		suburban
16		1a567,64	?		suburban
17		1a567,64	?		suburban
1		5a184,00	?		suburban
2		5a184,00	?		suburban
3		5a184,00	?		suburban
4		5a184,00	?		suburban
5		5a184,00	?		suburban
6		5a184,00	?		suburban
7		5a184,00	?		suburban
8		5a184,00	?		suburban
9		5a184,00	?		suburban

2. Для создания нового рейса на поезд.

```

Passenger=# create or replace function passenger.create_several_carriages(
Passenger(#   id_train integer,
Passenger(#   _price money,
Passenger(#   _departure_time timestamp,
Passenger(#   _arrival_time timestamp
Passenger(# ) returns integer as $create_several_carriages$
Passenger$# declare
Passenger$#     number_of_seats_1 integer;
Passenger$#     number_of_seats_2 integer;
Passenger$#     id_carriage_1 integer;
Passenger$#     id_carriage_2 integer;
Passenger$#     id_several_carriage integer;
Passenger$# begin
Passenger$#     id_carriage_1 = random()*(16);
Passenger$#     number_of_seats_1 = (
Passenger$#         select
Passenger$#             carriages_directory.number_of_seats
Passenger$#         from
Passenger$#             passenger.carriages_directory,
Passenger$#             passenger.carriages
Passenger$#         where
Passenger$#             carriages.carriage_id = id_carriage_1
Passenger$#             and carriages.carriage_directory_id = carriages_directory.carriage_directory_id
Passenger$#     );
Passenger$#     id_carriage_2 = random()*(16);
Passenger$#     number_of_seats_2 = (
Passenger$#         select
Passenger$#             carriages_directory.number_of_seats
Passenger$#         from
Passenger$#             passenger.carriages_directory,
Passenger$#             passenger.carriages
Passenger$#         where
Passenger$#             carriages.carriage_id = id_carriage_2
Passenger$#             and carriages.carriage_directory_id = carriages_directory.carriage_directory_id

```

```

Passenger$$
Passenger$$ if number_of_seats_1 is null then number_of_seats_1 = 0;
Passenger$$ end if;
Passenger$$
Passenger$$ if number_of_seats_2 is null then number_of_seats_2 = 0;
Passenger$$ end if;
Passenger$$
Passenger$$ id_several_carriage = nextval('passenger.several_carriages_id_seq'::regclass);
Passenger$$ insert into passenger.several_carriages(several_carriages_id)
Passenger$$ values (id_several_carriage);
Passenger$$
Passenger$$ insert into passenger.timetable(timetable_id, train_id, status, departure_time, arrival_time, several_carriages_id)
Passenger$$ values (nextval('passenger.carriage_id_seq'::regclass), id_train, 'scheduled', departure_time, arrival_time, id_several_carriages);
Passenger$$
Passenger$$ insert into railways.carriages(several_carriages_id, carriage_id, timetable_id, carriage_order_number)
Passenger$$ values (nextval('passenger.carriage_id_seq'::regclass), id_carriage_1, id_several_carriages, 1);
Passenger$$
Passenger$$ insert into railways.carriages(several_carriages_id, carriage_id, timetable_id, carriage_order_number)
Passenger$$ values (nextval('passenger.carriage_id_seq'::regclass), id_carriage_2, id_several_carriages, 2);
Passenger$$
Passenger$$ for i in 1..number_of_seats_1 loop
Passenger$$ insert into passenger.seats(seat_id, seat_number, seat_status, price, several_carriages_id)
Passenger$$ values (nextval('passenger.seat_id_seq'::regclass), 1, true, _price, id_carriage_1);
Passenger$$ end loop;
Passenger$$
Passenger$$ for i in 1..number_of_seats_2 loop
Passenger$$ insert into passenger.seats(seat_id, seat_number, seat_status, price, several_carriages_id)
Passenger$$ values (nextval('passenger.seat_id_seq'::regclass), 1, true, _price, id_carriage_2);
Passenger$$ end loop;
Passenger$$
Passenger$$ return number_of_seats_1 + number_of_seats_2;
Passenger$$ end;
Passenger$$ $create_several_carriages$ language plpgsql;
CREATE FUNCTION

```

Создание поезда:

```

Passenger=# select passenger.create_several_carriages(21, 4621::money, '2024-03-03 04:05:06'::timestamp, '2024-03-05 11:25:46'::timestamp);
create_several_carriages

```

Выгрузка расписания (до и после):

```

Passenger=# select *
Passenger-# from
Passenger-#     passenger.timetable
Passenger-# where timetable.train_id = 1;
 timetable_id | train_id | status | departure_time | arrival_time
-----+-----+-----+-----+-----
          102 |         1 | scheduled | 2024-01-26 03:43:25 | 2024-01-28 08:54:32
           51 |         1 | scheduled | 2024-01-29 09:43:25 | 2024-01-30 06:41:32
        1001 |         1 | scheduled | 2024-01-31 03:43:25 | 2024-02-02 08:54:32
           32 |         1 | scheduled | 2024-02-10 11:27:25 | 2024-02-11 08:41:32
           12 |         1 | scheduled | 2024-02-23 11:23:41 | 2024-02-24 08:54:32
           64 |         1 | scheduled | 2024-02-23 09:43:25 | 2024-02-25 08:36:32
            1 |         1 | scheduled | 2024-03-06 04:05:57 | 2024-03-09 04:05:57
(7 ÷)

```

```

Passenger=# select
Passenger-#     *
Passenger-# from
Passenger-#     passenger.timetable
Passenger-# where
Passenger-# timetable.train_id = 1
Passenger-# ;
 timetable_id | train_id | status | departure_time | arrival_time
-----+-----+-----+-----+-----
          102 |         1 | scheduled | 2024-01-26 03:43:25 | 2024-01-28 08:54:32
           51 |         1 | scheduled | 2024-01-29 09:43:25 | 2024-01-30 06:41:32
        1001 |         1 | scheduled | 2024-01-31 03:43:25 | 2024-02-02 08:54:32
           32 |         1 | scheduled | 2024-02-10 11:27:25 | 2024-02-11 08:41:32
           12 |         1 | scheduled | 2024-02-23 11:23:41 | 2024-02-24 08:54:32
           64 |         1 | scheduled | 2024-02-23 09:43:25 | 2024-02-25 08:36:32
            1 |         1 | scheduled | 2024-03-06 04:05:57 | 2024-03-09 04:05:57
           21 |         1 | scheduled | 2024-03-03 04:05:06 | 2024-03-05 11:25:46
(8 ÷)

```

3. Для формирования общей выручки по продаже билетов за сутки.

```
Passenger=# create or replace function passenger.get_money_for_day(  
Passenger(# data timestamp  
Passenger(# ) returns money as $get_money_for_day$  
Passenger$$ declare  
Passenger$$ get_money_plus money;  
Passenger$$ get_money_minus money;  
Passenger$$ get_money money;  
Passenger$$ begin  
Passenger$$ get_money_plus = (  
Passenger$$ select sum(seats.price)  
Passenger$$ from passenger.seats, passenger.tickets  
Passenger$$ where  
Passenger$$ tickets.buying_time <= date_trunc('day', data + interval '1 day')  
Passenger$$ and tickets.buying_time >= date_trunc('day', data)  
Passenger$$ and tickets.seat_id = seats.seat_id  
Passenger$$ and tickets.status = 'bought'  
Passenger$$ );  
Passenger$$ get_money_minus = (  
Passenger$$ select sum(seats.price)  
Passenger$$ from passenger.seats, passenger.tickets  
Passenger$$ where  
Passenger$$ tickets.buying_time <= date_trunc('day', data + interval '1 day')  
Passenger$$ and tickets.buying_time >= date_trunc('day', data)  
Passenger$$ and tickets.seat_id = seats.seat_id  
Passenger$$ and tickets.status = 'returned'  
Passenger$$ );  
Passenger$$ if get_money_plus::numeric is null then get_money_plus = 0::money;  
Passenger$$ end if;  
Passenger$$ if get_money_minus::numeric is null then get_money_minus = 0::money;  
Passenger$$ end if;  
Passenger$$ get_money = (get_money_plus::numeric - get_money_minus::numeric);  
Passenger$$ return get_money;  
Passenger$$ end;  
Passenger$$ $get_money_for_day$ language plpgsql;
```

```
Passenger=# SELECT passenger.get_money_for_day('2024-01-14'::timestamp);  
get_money_for_day  
-----  
7 575,55 ?  
(1 строка)
```

```
Passenger=# select tickets.buying_time, tickets.seat_id, seats.price  
Passenger-# from passenger.tickets, passenger.seats  
Passenger-# where tickets.seat_id = seats.seat_id;  
buying_time | seat_id | price  
-----+-----+-----  
2024-01-12 00:00:00 | 1 | 5 124,00 ?  
2024-01-12 00:00:00 | 3 | 5 174,40 ?  
2024-01-12 00:00:00 | 2 | 4 123,00 ?  
2024-01-12 00:00:00 | 0 | 3 687,55 ?  
2024-01-12 00:00:00 | 4 | 3 687,55 ?  
2024-01-14 10:04:52 | 5 | 3 687,55 ?  
2024-01-14 20:10:34 | 6 | 3 888,00 ?  
(7 строк)
```

Создание триггера (триггер проверяет статус места, связанного с новым билетом. Если статус места не пустой, вставка строки будет разрешена. В противном случае (если статус места пустой или не существует), вставка строки будет отменена)

```
Passenger=# create or replace function fn_check_is_empty_status() returns trigger as $fn_check_is_empty_status$
Passenger$$ begin
Passenger$$   if (select distinct seats.seat_status
Passenger$$       from passenger.seats, passenger.tickets
Passenger$$       where new.seat = seats.seat_id) then return new;
Passenger$$   else
Passenger$$       return null;
Passenger$$   end if;
Passenger$$ end;
Passenger$$ $fn_check_is_empty_status$ language plpgsql;
CREATE FUNCTION
```

```
Passenger=# create trigger check_is_empty_status before insert on passenger.tickets
Passenger=# for each row execute procedure fn_check_is_empty_status();
CREATE TRIGGER
```

```
Passenger=# select * from passenger.seats where seat_id = 6;
 seat_id | seat_number | price   | seat_type | tier | floor | seat_status | several_carriages_id
-----+-----+-----+-----+-----+-----+-----+-----
      6 |          3 | 3 888,00 ? | compartment |    1 |     1 | t           |                  0
(1 строка)
```

```
Passenger=# insert into passenger.tickets(ticket_id, passenger_id, offices_id, status, buying_time, seat_id, buying_format)
Passenger=# values (nextval('passenger.ticket_id_seq'::regclass), 6, 2, 1, 'bought', '2024-01-14 20:10:34', 6, 'online');
INSERT 0 0
```


Вывод

В ходе лабораторной работы была освоена работа с процедурами и триггерами.