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: Бунос М.В.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

1. Создать процедуры согласно индивидуальному заданию

1. Выполнить списание автомобилей, выпущенных ранее заданного года.

```
CREATE OR REPLACE PROCEDURE WriteOffCars(before_year INTEGER)
LANGUAGE plpgsql
AS $$
BEGIN
  DELETE FROM cars WHERE issue_year < before_year;
END;
$$;
```

До:

```
 id | registration_number | mileage | issue_year | engine_number | body_number |  price  | last_inspection | special_mark
s | return_mark | model_id
----+---------------------+---------+------------+---------------+-------------+---------+-----------------+-------------
--+-------------+----------
  1 | O0010O77            |   10000 |       2021 | EN12345       | BN12345     |  500000 | 2023-01-01      |
  |           1 |        1
  2 | A222AA177           |    5000 |       2022 | EN54321       | BN54321     |  400000 | 2023-01-02      |
  |           1 |        2
  3 | P333PP77            |   15000 |       2020 | EN67890       | BN67890     |  600000 | 2023-01-03      |
  |           0 |        3
  4 | X228EP98            |   15000 |       2020 | EN66666       | BN77777     | 1300000 | 2023-01-01      |
  |           1 |        1
(4 rows)
```

После:

```
g1phy=> CALL WriteOffCars(2021);
CALL
g1phy=> select * from cars;
 id | registration_number | mileage | issue_year | engine_number | body_number | price  | last_inspection | special_marks
 | return_mark | model_id
----+---------------------+---------+------------+---------------+-------------+--------+-----------------+--------------
--+-------------+----------
  1 | O0010O77            |   10000 |       2021 | EN12345       | BN12345     | 500000 | 2023-01-01      |
  |           1 |        1
  2 | A222AA177           |    5000 |       2022 | EN54321       | BN54321     | 400000 | 2023-01-02      |
  |           1 |        2
(2 rows)
```

2. Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.

```
CREATE OR REPLACE PROCEDURE RentCarWithDiscount(
  client_id INT,
  car_id INT,
  rental_period INT,
  deposit DECIMAL,
```

```
  insurance_id INT,
  employee_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
  base_price DECIMAL;
  discount_rate DECIMAL;
  final_price DECIMAL;
BEGIN
  SELECT price INTO base_price
  FROM rent_prices
  WHERE model_id = (SELECT model_id FROM cars WHERE id = car_id)
    AND start_datetime <= CURRENT_TIMESTAMP
    AND (end_datetime IS NULL OR end_datetime >= CURRENT_TIMESTAMP)
  LIMIT 1;

  SELECT discount INTO discount_rate
  FROM clients
  WHERE id = client_id;

  final_price := base_price * rental_period * (1 - discount_rate / 100);

  INSERT INTO contracts (
    given_datetime,
    result_price,
    deposit,
    returned_deposit,
    payment_status,
    common_status,
    car_id,
    insurance_id,
    employee_id,
    client_id
  ) VALUES (
    CURRENT_TIMESTAMP,
    final_price,
    deposit,
    FALSE,
    FALSE,
    1,
    car_id,
    insurance_id,
    employee_id,
    client_id
  );

  RAISE NOTICE 'The final rental price is: %', final_price;
END;
$$;
```

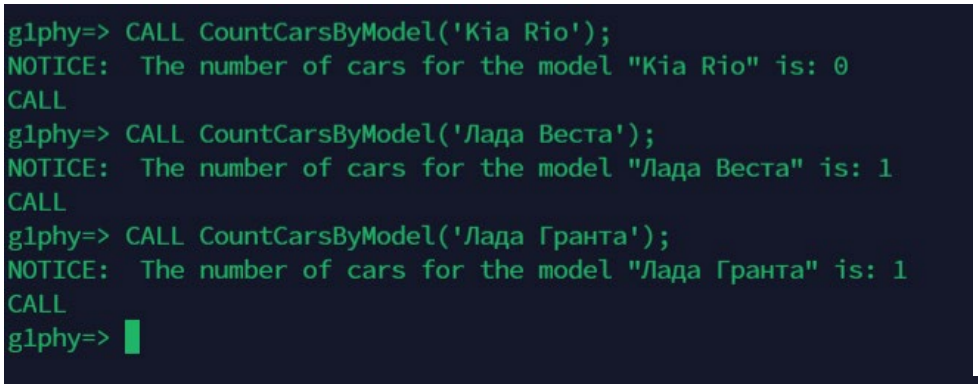
```
g1phy=> CALL RentCarWithDiscount(1, 2, 10, 300.00, 1, 1);
NOTICE:  The final rental price is: 99000.000000000000000000
CALL
```

```
 id |       given_datetime       |   returned_datetime    | result_price | deposit | returned_deposit | payment_status
 | common_status | car_id | insurance_id | employee_id | client_id
----+----------------------------+------------------------+--------------+---------+------------------+---------------
--+---------------+--------+--------------+-------------+----------
  1 | 2023-05-10 10:00:00+03       | 2023-11-09 08:00:00+03 |        25000 |    5000 | f                | f
 |             0 |      1 |            1 |           1 |        1
  2 | 2023-05-12 12:00:00+03       | 2023-11-09 12:00:00+03 |        30000 |   10000 | f                | f
 |             0 |      1 |            2 |           2 |        2
  3 | 2023-05-14 14:00:00+03       | 2023-11-09 11:00:00+03 |        35000 |   15000 | f                | f
 |             1 |      1 |            3 |           1 |        3
  6 | 2023-12-07 22:01:03.73042+03 |                        |        99000 |     300 | f                | f
 |             1 |      2 |            1 |           1 |        1
  7 | 2023-12-07 22:01:09.851441+03 |                        |        99000 |     300 | f                | f
 |             1 |      2 |            1 |           1 |        1
(5 rows)
```

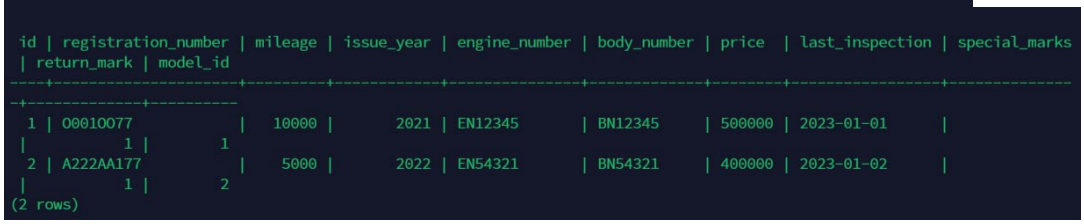
3. Для вычисления количества автомобилей заданной марки.

```
CREATE OR REPLACE PROCEDURE CountCarsByModel(model_name TEXT)
LANGUAGE plpgsql
AS $$
DECLARE
  car_count INT;
BEGIN
  SELECT COUNT(*)
  INTO car_count
  FROM cars
  JOIN car_models ON cars.model_id = car_models.id
  WHERE car_models.name = model_name;

  RAISE NOTICE 'The number of cars for the model "%" is: %', model_name,
car_count;
END;
$$;
```



```
g1phy=> CALL CountCarsByModel('Kia Rio');
NOTICE:  The number of cars for the model "Kia Rio" is: 0
CALL
g1phy=> CALL CountCarsByModel('Лада Веста');
NOTICE:  The number of cars for the model "Лада Веста" is: 1
CALL
g1phy=> CALL CountCarsByModel('Лада Гранта');
NOTICE:  The number of cars for the model "Лада Гранта" is: 1
CALL
g1phy=>
```



```
 id | registration_number | mileage | issue_year | engine_number | body_number | price  | last_inspection | special_marks
 | return_mark | model_id
----+---------------------+---------+------------+---------------+-------------+--------+-----------------+--------------
--+-------------+----------
  1 | O001OO77            |   10000 |       2021 | EN12345       | BN12345     | 500000 | 2023-01-01      |
 |           1 |        1
  2 | A222AA177           |    5000 |       2022 | EN54321       | BN54321     | 400000 | 2023-01-02      |
 |           1 |        2
(2 rows)
```

2. Создать необходимый триггер

Триггер будет автоматически обновлять статус автомобиля на "недоступен" после того, как контракт на его аренду будет создан. Это логичное действие, поскольку автомобиль, который уже арендован, не должен быть доступен для новых аренд до возврата.

```sql
CREATE OR REPLACE FUNCTION update_car_status()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE cars
  SET return_mark = 0
  WHERE id = NEW.car_id;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_car_status
AFTER INSERT ON contracts
FOR EACH ROW
EXECUTE FUNCTION update_car_status();
```

```
 id | registration_number | mileage | issue_year | engine_number | body_number | price  | last_inspection | special_marks
 | return_mark | model_id
----+---------------------+---------+------------+---------------+-------------+--------+-----------------+--------------
-+-------------+----------
  1 | O0010O77            |   10000 |       2021 | EN12345       | BN12345     | 500000 | 2023-01-01      |
 |           1 |        1
  2 | A222AA177           |    5000 |       2022 | EN54321       | BN54321     | 400000 | 2023-01-02      |
 |           1 |        2
(2 rows)
```

```
g1phy=> CALL RentCarWithDiscount(1, 2, 10, 300.00, 1, 1);
NOTICE:  The final rental price is: 99000.000000000000000000
CALL
```

```
 id | registration_number | mileage | issue_year | engine_number | body_number | price  | last_inspection | special_marks
 | return_mark | model_id
----+---------------------+---------+------------+---------------+-------------+--------+-----------------+--------------
-+-------------+----------
  1 | O0010O77            |   10000 |       2021 | EN12345       | BN12345     | 500000 | 2023-01-01      |
 |           1 |        1
  2 | A222AA177           |    5000 |       2022 | EN54321       | BN54321     | 400000 | 2023-01-02      |
 |           0 |        2
(2 rows)
```

Вывод

В ходе лабораторной работы были успешно созданы и протестированы процедуры и триггеры в системе управления базами данных PostgreSQL. Работа позволила овладеть практическими навыками программирования в SQL и пониманием механизма триггеров.