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»

Обучающийся Генне Константин Валерьевич
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

1 Цель работы

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

2 Практическое задание

Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)

Создать триггеры для индивидуальной БД согласно варианту:

Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

Дополнительные баллы - 3:

Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).

Указание. Работа выполняется в консоли SQL Shell (psql).

3 Выполнение работы

3.1 Процедуры и функции

1) Повышения цены деталей для автомобиля “Ford” на 10 %.

Процедура:

```
car_workshop=# CREATE OR REPLACE PROCEDURE increase_ford_details_price()
car_workshop-# LANGUAGE plpgsql
car_workshop-# AS $$
car_workshop$# BEGIN
car_workshop$#     UPDATE detail
car_workshop$#     SET price = price * 1.10
car_workshop$#     WHERE car_brand = 'Ford';
car_workshop$# END;
car_workshop$# $$;
CREATE PROCEDURE
```

Исходные данные:

```
car_workshop=# select * FROM detail d WHERE d.car_brand = 'Ford';
 id |         name          |  price  | car_brand | car_model | country_of_origin
----+-----------------------+---------+-----------+-----------+-------------------
 16 | Термостат             |    2100 | Ford      | Focus     | США
  7 | Амортизатор передний | 5175.00 | Ford      | Focus     | США
(2 строки)
```

Вызов процедуры:

```
car_workshop=# CALL increase_ford_details_price();
CALL
```

Результат выполнения:

```
car_workshop=# CALL increase_ford_details_price();
CALL
car_workshop=# select * FROM detail d WHERE d.car_brand = 'Ford';
 id |         name          |   price   | car_brand | car_model | country_of_origin
----+-----------------------+-----------+-----------+-----------+-------------------
 16 | Термостат             |   2310.00 | Ford      | Focus     | США
  7 | Амортизатор передний | 5692.5000 | Ford      | Focus     | США
(2 строки)
```

2) Для повышения разряда тех мастеров, которые отремонтировали больше 3 автомобилей.

Процедура:

```
car_workshop=# CREATE OR REPLACE PROCEDURE promote_mechanics()
car_workshop-# LANGUAGE plpgsql
car_workshop-# AS $$
car_workshop$# DECLARE
car_workshop$#     r RECORD;
car_workshop$#     max_category_number INTEGER;
car_workshop$#     current_category_number INTEGER;
car_workshop$#     new_category_number INTEGER;
car_workshop$#     new_category_id INTEGER;
car_workshop$# BEGIN
car_workshop$#     SELECT MAX(category_number::INTEGER) INTO max_category_number FROM category;
car_workshop$#
car_workshop$#     FOR r IN
car_workshop$#         SELECT e.service_number
car_workshop$#         FROM employee e
car_workshop$#         JOIN employee_position_period epp
car_workshop$#             ON e.service_number = epp.employee_service_number
car_workshop$#         JOIN position p
car_workshop$#             ON epp.id_position = p.id
car_workshop$#         WHERE epp.end_date IS NULL AND p.id = 1
car_workshop$#         AND (
car_workshop$#             SELECT COUNT(DISTINCT c.id)
car_workshop$#             FROM service_provided sp
car_workshop$#             JOIN contract c ON sp.id_contract = c.id
car_workshop$#             WHERE sp.employee_service_number = e.service_number
car_workshop$#         ) > 3
car_workshop$#     LOOP
car_workshop$#         SELECT c.category_number::INTEGER
car_workshop$#         INTO current_category_number
car_workshop$#         FROM category c
car_workshop$#         JOIN employee_category_period ecp ON c.id = ecp.id_category
car_workshop$#         WHERE ecp.employee_service_number = r.service_number AND ecp.end_date IS NULL;
car_workshop$#
car_workshop$#         IF current_category_number < max_category_number THEN
car_workshop$#             new_category_number := current_category_number + 1;
car_workshop$#
car_workshop$#             SELECT id INTO new_category_id
car_workshop$#             FROM category
car_workshop$#             WHERE category_number::INTEGER = new_category_number;
car_workshop$#
car_workshop$#
car_workshop$#             INSERT INTO employee_category_period(
car_workshop$#                 employee_service_number,
car_workshop$#                 id_category,
car_workshop$#                 start_date,
car_workshop$#                 end_date
car_workshop$#             ) VALUES (
car_workshop$#                 r.service_number,
car_workshop$#                 new_category_id,
car_workshop$#                 CURRENT_DATE,
car_workshop$#                 NULL
car_workshop$#             );
car_workshop$#         END IF;
car_workshop$#     END LOOP;
car_workshop$# END;
car_workshop$# $$;
CREATE PROCEDURE
```

Исходные данные (на первом скриншоте перечислены номера всех механиков, кто отремонтировал более 3 автомобилей; на втором скриншоте представлен фрагмент таблицы employee_category_period, в которой перечислены механики и их разряды):

```
car_workshop=# SELECT e.service_number
car_workshop-#         FROM employee e
car_workshop-#         JOIN employee_position_period epp
car_workshop-#             ON e.service_number = epp.employee_service_number
car_workshop-#         JOIN position p
car_workshop-#             ON epp.id_position = p.id
car_workshop-#         WHERE epp.end_date IS NULL AND p.id = 1
car_workshop-#         AND (
car_workshop(#             SELECT COUNT(DISTINCT c.id)
car_workshop(#             FROM service_provided sp
car_workshop(#             JOIN contract c ON sp.id_contract = c.id
car_workshop(#             WHERE sp.employee_service_number = e.service_number
car_workshop(#         ) > 3;
 service_number
----------------
              2
              4
              5
              6
              7
              9
             11
             12
             13
             15
             16
             17
             18
             19
             20
             21
             22
             23
             25
             26
             27
             29
             30
             31
             32
             33
             34
             35
             36
             37
             39
             43
             44
             45
             49
             50
(36 строк)
```

```
car_workshop=# SELECT * FROM employee_category_period;
 id | employee_service_number | id_category | start_date |  end_date
----+-------------------------+-------------+------------+------------
  1 |                       1 |           3 | 2020-07-09 | 2021-07-09
  2 |                       1 |           4 | 2021-07-10 | 2021-09-02
  3 |                       2 |           4 | 2021-05-15 |
  4 |                       3 |           3 | 2023-01-16 | 2024-01-16
  5 |                       3 |           4 | 2024-01-17 | 2025-01-17
  6 |                       3 |           5 | 2025-01-18 | 2025-01-22
  7 |                       4 |           4 | 2021-11-29 |
  8 |                       5 |           3 | 2022-06-13 | 2023-06-13
  9 |                       5 |           4 | 2023-06-14 |
```

Результат выполнения (например, был повышен разряд механика с номером 3):

```
car_workshop=# CALL promote_mechanics();
CALL
car_workshop=# SELECT * FROM employee_category_period;
 id  | employee_service_number | id_category | start_date |  end_date
-----+-------------------------+-------------+------------+------------
   1 |                       1 |           3 | 2020-07-09 | 2021-07-09
   2 |                       1 |           4 | 2021-07-10 | 2021-09-02
   4 |                       3 |           3 | 2023-01-16 | 2024-01-16
   5 |                       3 |           4 | 2024-01-17 | 2025-01-17
   6 |                       3 |           5 | 2025-01-18 | 2025-01-22
   8 |                       5 |           3 | 2022-06-13 | 2023-06-13
  11 |                       7 |           3 | 2020-07-12 | 2022-07-12
  13 |                       8 |           4 | 2020-09-27 | 2020-10-30
  15 |                      10 |           4 | 2023-01-04 | 2023-04-01
  17 |                      12 |           3 | 2021-06-07 | 2022-06-07
  20 |                      14 |           3 | 2022-03-15 | 2023-03-15
  21 |                      14 |           4 | 2023-03-16 | 2024-05-25
  24 |                      17 |           3 | 2021-08-10 | 2022-08-10
  27 |                      19 |           3 | 2021-04-13 | 2022-04-13
  32 |                      23 |           3 | 2021-01-19 | 2022-01-19
  34 |                      24 |           3 | 2021-02-10 | 2022-02-10
  35 |                      24 |           4 | 2022-02-11 | 2023-05-12
  37 |                      26 |           3 | 2021-04-02 | 2022-04-02
  40 |                      28 |           3 | 2022-05-30 | 2022-11-30
  41 |                      28 |           4 | 2022-12-01 | 2023-09-21
  46 |                      33 |           3 | 2022-01-11 | 2023-01-11
  49 |                      35 |           3 | 2021-08-02 | 2022-08-02
  53 |                      38 |           3 | 2020-02-16 | 2021-02-16
  54 |                      38 |           4 | 2021-02-17 | 2022-04-07
  56 |                      40 |           3 | 2020-03-17 | 2021-03-17
  57 |                      40 |           4 | 2021-03-18 | 2022-01-19
  58 |                      41 |           3 | 2020-11-02 | 2021-11-02
  59 |                      41 |           4 | 2021-11-03 | 2022-11-03
  60 |                      41 |           5 | 2022-11-04 | 2023-09-21
  61 |                      42 |           3 | 2020-09-12 | 2021-09-12
  62 |                      42 |           4 | 2021-09-13 | 2022-04-29
  66 |                      46 |           3 | 2022-03-15 | 2023-03-15
  67 |                      46 |           4 | 2023-03-16 | 2025-01-02
  68 |                      47 |           4 | 2022-04-08 | 2022-11-07
  69 |                      48 |           3 | 2020-01-30 | 2021-01-30
  70 |                      48 |           4 | 2021-01-31 | 2022-12-16
  72 |                      50 |           3 | 2021-06-03 | 2022-06-03
   3 |                       2 |           4 | 2021-05-15 | 2025-05-20
  74 |                       2 |           5 | 2025-05-21 |
   7 |                       4 |           4 | 2021-11-29 | 2025-05-20
  75 |                       4 |           5 | 2025-05-21 |
   9 |                       5 |           4 | 2023-06-14 | 2025-05-20
```

3) Сколько автомобилей отремонтировал каждый механик за истекший квартал.

Функция:

```
car_workshop=# CREATE OR REPLACE FUNCTION count_cars_repaired_last_quarter()
car_workshop-# RETURNS TABLE (service_number INTEGER, mechanic_name CHARACTER VARYING, cars_repaired BIGINT)
car_workshop-# LANGUAGE plpgsql
car_workshop-# AS $$
car_workshop$# BEGIN
car_workshop$#     RETURN QUERY
car_workshop$#     SELECT e.service_number, np.full_name, COUNT(DISTINCT c.id)
car_workshop$#     FROM employee e
car_workshop$#     JOIN service_provided sp ON e.service_number = sp.employee_service_number
car_workshop$#  JOIN natural_person np ON e.id_n_p = np.id
car_workshop$#     JOIN contract c ON sp.id_contract = c.id
car_workshop$#     WHERE sp.actual_date_completion_repair >= date_trunc('quarter', CURRENT_DATE - INTERVAL '3 months')
car_workshop$#       AND sp.actual_date_completion_repair < date_trunc('quarter', CURRENT_DATE)
car_workshop$#     GROUP BY e.service_number, np.id;
car_workshop$# END;
car_workshop$# $$;
CREATE FUNCTION
```

Результат выполнения:

```
car_workshop=# SELECT * FROM count_cars_repaired_last_quarter();
 service_number |          mechanic_name          | cars_repaired
----------------+---------------------------------+---------------
              3 | Александров Макар Александрович |             1
              4 | Сидоров Александр Родионович    |             2
              5 | Шмидт Андрей Иванович           |             2
              7 | Скоробогатов Иван Родионович    |             3
              8 | Андреев Владимир Владимирович   |             1
             10 | Александров Макар Александрович |             1
             11 | Васильев Егор Алексеевич        |             1
             15 | Сидоров Андрей Иванович         |             1
             16 | Васильев Владимир Егорович      |             1
             17 | Раскольников Макар Васильевич   |             1
             18 | Смирнов Макар Макарович         |             1
             22 | Иванов Алексей Родионович       |             1
             23 | Шмидт Макар Макарович           |             1
             24 | Смирнов Алексей Егорович        |             1
             26 | Чацкий Марат Макарович          |             2
             27 | Сидоров Иван Маратович          |             2
             28 | Скоробогатов Иван Александрович |             1
             30 | Васильев Марат Егорович         |             2
             32 | Александров Александр Маратович |             1
             35 | Скоробогатов Макар Владимирович |             1
             36 | Андреев Егор Андреевич          |             2
             37 | Васильев Алексей Родионович     |             1
             38 | Раскольников Егор Егорович      |             2
             40 | Шмидт Егор Иванович             |             1
             42 | Скоробогатов Алексей Макарович  |             1
             43 | Смирнов Василий Иванович        |             1
             45 | Скоробогатов Егор Александрович |             3
             46 | Александров Василий Маратович   |             1
             50 | Раскольников Иван Егорович      |             1
             51 | Раскольников Андрей Макарович   |             1
             53 | Чацкий Егор Родионович          |             1
             57 | Иванов Макар Алексеевич         |             1
             59 | Андреев Василий Васильевич      |             1
             60 | Скоробогатов Егор Егорович      |             1
             65 | Раскольников Андрей Алексеевич  |             1
             67 | Раскольников Макар Иванович     |             1
             70 | Скоробогатов Егор Владимирович  |             1
             71 | Андреев Андрей Андреевич        |             1
             78 | Шмидт Василий Макарович         |             1
(39 строк)
```

3.2 Триггеры

1) Триггер на обновление предыдущей записи с разрядом механика при присвоении ему нового разряда

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION close_previous_category()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# BEGIN
car_workshop$#     UPDATE employee_category_period
car_workshop$#     SET end_date = NEW.start_date - INTERVAL '1 day'
car_workshop$#     WHERE employee_service_number = NEW.employee_service_number AND end_date IS NULL;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_close_previous_category
car_workshop-# BEFORE INSERT ON employee_category_period
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE close_previous_category();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM employee_category_period;
 id | employee_service_number | id_category | start_date |  end_date
----+-------------------------+-------------+------------+------------
  1 |                       1 |           3 | 2020-07-09 | 2021-07-09
  2 |                       1 |           4 | 2021-07-10 | 2021-09-02
  3 |                       2 |           4 | 2021-05-15 |
  4 |                       3 |           3 | 2023-01-16 | 2024-01-16
  5 |                       3 |           4 | 2024-01-17 | 2025-01-17
  6 |                       3 |           5 | 2025-01-18 | 2025-01-22
  7 |                       4 |           4 | 2021-11-29 |
```

Результат выполнения (триггер автоматически заполнил дату end_date на основании новой записи):

```
car_workshop=# CALL promote_mechanics();
CALL
car_workshop=# SELECT * FROM employee_category_period;
 id  | employee_service_number | id_category | start_date |  end_date
-----+-------------------------+-------------+------------+------------
   1 |                       1 |           3 | 2020-07-09 | 2021-07-09
   2 |                       1 |           4 | 2021-07-10 | 2021-09-02
   4 |                       3 |           3 | 2023-01-16 | 2024-01-16
   5 |                       3 |           4 | 2024-01-17 | 2025-01-17
   6 |                       3 |           5 | 2025-01-18 | 2025-01-22
   8 |                       5 |           3 | 2022-06-13 | 2023-06-13
  11 |                       7 |           3 | 2020-07-12 | 2022-07-12
  13 |                       8 |           4 | 2020-09-27 | 2020-10-30
  15 |                      10 |           4 | 2023-01-04 | 2023-04-01
  17 |                      12 |           3 | 2021-06-07 | 2022-06-07
  20 |                      14 |           3 | 2022-03-15 | 2023-03-15
  21 |                      14 |           4 | 2023-03-16 | 2024-05-25
  24 |                      17 |           3 | 2021-08-10 | 2022-08-10
  27 |                      19 |           3 | 2021-04-13 | 2022-04-13
  32 |                      23 |           3 | 2021-01-19 | 2022-01-19
  34 |                      24 |           3 | 2021-02-10 | 2022-02-10
  35 |                      24 |           4 | 2022-02-11 | 2023-05-12
  37 |                      26 |           3 | 2021-04-02 | 2022-04-02
  40 |                      28 |           3 | 2022-05-30 | 2022-11-30
  41 |                      28 |           4 | 2022-12-01 | 2023-09-21
  46 |                      33 |           3 | 2022-01-11 | 2023-01-11
  49 |                      35 |           3 | 2021-08-02 | 2022-08-02
  53 |                      38 |           3 | 2020-02-16 | 2021-02-16
  54 |                      38 |           4 | 2021-02-17 | 2022-04-07
  56 |                      40 |           3 | 2020-03-17 | 2021-03-17
  57 |                      40 |           4 | 2021-03-18 | 2022-01-19
  58 |                      41 |           3 | 2020-11-02 | 2021-11-02
  59 |                      41 |           4 | 2021-11-03 | 2022-11-03
  60 |                      41 |           5 | 2022-11-04 | 2023-09-21
  61 |                      42 |           3 | 2020-09-12 | 2021-09-12
  62 |                      42 |           4 | 2021-09-13 | 2022-04-29
  66 |                      46 |           3 | 2022-03-15 | 2023-03-15
  67 |                      46 |           4 | 2023-03-16 | 2025-01-02
  68 |                      47 |           4 | 2022-04-08 | 2022-11-07
  69 |                      48 |           3 | 2020-01-30 | 2021-01-30
  70 |                      48 |           4 | 2021-01-31 | 2022-12-16
  72 |                      50 |           3 | 2021-06-03 | 2022-06-03
   3 |                       2 |           4 | 2021-05-15 | 2025-05-20
  74 |                       2 |           5 | 2025-05-21 |
   7 |                       4 |           4 | 2021-11-29 | 2025-05-20
  75 |                       4 |           5 | 2025-05-21 |
   9 |                       5 |           4 | 2023-06-14 | 2025-05-20
```

2) Триггер для проверки даты начала владения автомобилем (дата начала владения не может быть меньше даты выпуска автомобиля)

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION check_start_date_ownership()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     car_release_date date;
car_workshop$# BEGIN
car_workshop$#     SELECT release_year INTO car_release_date FROM car WHERE state_number = NEW.state_number_car;
car_workshop$#
car_workshop$#     IF NEW.start_date_ownership < car_release_date THEN
car_workshop$#         RAISE EXCEPTION 'The start date of ownership cannot be earlier than the release date of the car';
car_workshop$#     END IF;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_check_start_date_ownership
car_workshop-# BEFORE INSERT ON client_car
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE check_start_date_ownership();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM car WHERE state_number = 'C930CX78';
 state_number | brand | model |   colour   | release_year | power
--------------+-------+-------+------------+--------------+-------
 C930CX78     | Audi  | Focus | Коричневый | 2013-08-08   |   140
(1 строка)
```

Результат выполнения (попытка создать новую запись с некорректной датой начала владения автомобилем завершилась ошибкой):

```
car_workshop=# INSERT INTO client_car (id_client, state_number_car, start_date_ownership, end_date_ownership) VALUES (1900, 'C930CX78', '2012-08-01', null);
ОШИБКА:  The start date of ownership cannot be earlier than the release date of the car
КОНТЕКСТ:  функция PL/pgSQL check_start_date_ownership(), строка 8, оператор RAISE
```

3) Триггер для проверки соответствия количества деталей в заказе количеству по плану

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION check_detail_quantity()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     plan_quantity integer;
car_workshop$# BEGIN
car_workshop$#     SELECT sc.num_of_details_plan
car_workshop$#     INTO plan_quantity
car_workshop$#     FROM service_composition sc, service_provided sp
car_workshop$#     WHERE NEW.id_service_provided = sp.id AND sp.id_service = sc.id_service AND sc.id_detail = NEW.id_detail;
car_workshop$#
car_workshop$#     IF COALESCE(NEW.quantity_from_company, 0) + COALESCE(NEW.quantity_from_client, 0) <> plan_quantity THEN
car_workshop$#         RAISE EXCEPTION 'The total number of details does not match the plan';
car_workshop$#     END IF;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_check_detail_quantity
car_workshop-# BEFORE INSERT ON detail_in_order
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE check_detail_quantity();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM service_composition WHERE id_service = 2 AND id_detail = 5;
 id | id_service | id_detail | num_of_details_plan
----+------------+-----------+---------------------
  1 |          2 |         5 |                   1
(1 строка)
```

Результат выполнения (попытка создать новую запись с некорректным количеством деталей завершилась ошибкой):

```
car_workshop=# INSERT INTO detail_in_order (id_detail, quantity_from_company, quantity_from_client, id_service_provided) VALUES (5, 1, 1, 7);
ОШИБКА:  The total number of details does not match the plan
КОНТЕКСТ:  функция PL/pgSQL check_detail_quantity(), строка 11, оператор RAISE
```

4) Триггер на автоматический подсчёт стоимости деталей для оказания конкретной услуги

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION calculate_detail_cost()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     unit_price numeric;
car_workshop$# BEGIN
car_workshop$#     SELECT price INTO unit_price FROM detail WHERE id = NEW.id_detail;
car_workshop$#
car_workshop$#     NEW.detail_cost := COALESCE(NEW.quantity_from_company, 0) * unit_price;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_calculate_detail_cost
car_workshop-# BEFORE INSERT ON detail_in_order
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE calculate_detail_cost();
CREATE TRIGGER
```

Результат выполнения (стоимость автоматически посчиталась, и она равна 0, так как все детали предоставляет клиент):

```
car_workshop=# INSERT INTO detail_in_order (id_detail, quantity_from_company, quantity_from_client, id_service_provided) VALUES (5, 0, 1, 7);
INSERT 0 1
car_workshop=# SELECT * FROM detail_in_order WHERE id = (SELECT MAX(id) FROM detail_in_order);
  id  | id_detail | quantity_from_company | quantity_from_client | comment | id_service_provided | detail_cost
------+-----------+-----------------------+----------------------+---------+---------------------+-------------
 2004 |         5 |                     0 |                    1 |         |                   7 |           0
(1 строка)
```

5) Триггер для проверки соответствия даты завершения ремонта по контракту дате завершения последней оказываемой услуги

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION check_contract_completion_date()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     latest_date date;
car_workshop$# BEGIN
car_workshop$#     SELECT MAX(actual_date_completion_repair)
car_workshop$#     INTO latest_date
car_workshop$#     FROM service_provided
car_workshop$#     WHERE id_contract = NEW.id;
car_workshop$#
car_workshop$#     IF NEW.actual_date_completion_repair IS NOT NULL AND NEW.actual_date_completion_repair <> latest_date THEN
car_workshop$#         RAISE EXCEPTION 'The contract repair completion date should be equal to the latest date among the services';
car_workshop$#     END IF;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_check_contract_completion_date
car_workshop-# BEFORE UPDATE ON contract
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE check_contract_completion_date();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM contract WHERE id = 3797;
  id  | service_number_employee | id_workshop | id_client_car | actual_date_completion_repair | date_order | date_acceptance_for_repair | sheduled_repair_co
mpletion_date
------+-------------------------+-------------+---------------+-------------------------------+------------+----------------------------+-------------------
--------------
 3797 |                      85 |           5 |           575 |                               | 2025-04-30 | 2025-04-30                 | 2025-04-30
(1 строка)
```

```
car_workshop=# SELECT * FROM service_provided WHERE id_contract = 3797;
  id  | id_service | id_contract | employee_service_number | date_acceptance_for_repair | sheduled_repair_completion_date | actual_date_completion_repair |
service_cost
------+------------+-------------+-------------------------+----------------------------+---------------------------------+-------------------------------+-
-------------
 2001 |         11 |        3797 |                       5 | 2025-04-30                 | 2025-04-30                      | 2025-04-30                    |
        8500
 2002 |         30 |        3797 |                       5 | 2025-04-30                 | 2025-05-01                      | 2025-05-01                    |
        2000
(2 строки)
```

Результат выполнения:

```
car_workshop=# UPDATE contract
car_workshop-# SET actual_date_completion_repair = '2025-04-30'
car_workshop-# WHERE id = 3797;
ОШИБКА:  The contract repair completion date should be equal to the latest date among the services
КОНТЕКСТ:  функция PL/pgSQL check_contract_completion_date(), строка 11, оператор RAISE
```

6) Триггер для проверки, что дата взятия в ремонт в записи предоставляемой услуги не меньше дате взятия в ремонт, указанной в контракте

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION check_service_acceptance_date()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     contract_accept_date date;
car_workshop$# BEGIN
car_workshop$#     SELECT date_acceptance_for_repair INTO contract_accept_date
car_workshop$#     FROM contract
car_workshop$#     WHERE id = NEW.id_contract;
car_workshop$#
car_workshop$#     IF NEW.date_acceptance_for_repair < contract_accept_date THEN
car_workshop$#         RAISE EXCEPTION 'The date of acceptance of the service cannot be earlier than the date of acceptance of the contract';
car_workshop$#     END IF;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_check_service_acceptance_date
car_workshop-# BEFORE INSERT ON service_provided
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE check_service_acceptance_date();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM contract WHERE id = 3797;
  id  | service_number_employee | id_workshop | id_client_car | actual_date_completion_repair | date_order | date_acceptance_for_repair | sheduled_repair_co
mpletion_date
------+-------------------------+-------------+---------------+-------------------------------+------------+----------------------------+-------------------
--------------
 3797 |                      85 |           5 |           575 |                               | 2025-04-30 | 2025-04-30                 | 2025-04-30
(1 строка)
```

Результат выполнения:

```
car_workshop=# INSERT INTO service_provided (id_service, id_contract, employee_service_number, date_acceptance_for_repair) VALUES (2, 3797, 5, '2025-04-29')
;
ОШИБКА:  The date of acceptance of the service cannot be earlier than the date of acceptance of the contract
КОНТЕКСТ:  функция PL/pgSQL check_service_acceptance_date(), строка 10, оператор RAISE
```

7) Триггер для проверки соответствия номера автомастерской, к которой прикреплён сотрудник, номеру автомастерской, в которой оказывается услуга

Триггер:

```
car_workshop=# CREATE OR REPLACE FUNCTION check_employee_workshop()
car_workshop-# RETURNS TRIGGER AS $$
car_workshop$# DECLARE
car_workshop$#     contract_workshop integer;
car_workshop$#     employee_workshop integer;
car_workshop$# BEGIN
car_workshop$#     SELECT id_workshop INTO contract_workshop FROM contract WHERE id = NEW.id_contract;
car_workshop$#     SELECT id_workshop INTO employee_workshop FROM employee WHERE service_number = NEW.employee_service_number;
car_workshop$#
car_workshop$#     IF contract_workshop <> employee_workshop THEN
car_workshop$#         RAISE EXCEPTION 'The employee works at a car workshop at a different address';
car_workshop$#     END IF;
car_workshop$#
car_workshop$#     RETURN NEW;
car_workshop$# END;
car_workshop$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_workshop=# CREATE TRIGGER trg_check_employee_workshop
car_workshop-# BEFORE INSERT ON service_provided
car_workshop-# FOR EACH ROW
car_workshop-# EXECUTE PROCEDURE check_employee_workshop();
CREATE TRIGGER
```

Исходные данные:

```
car_workshop=# SELECT * FROM employee WHERE service_number = 1;
 service_number | id_n_p | id_workshop
----------------+--------+-------------
              1 |      1 |           1
(1 строка)
```

```
car_workshop=# SELECT * FROM contract WHERE id = 3797;
  id  | service_number_employee | id_workshop | id_client_car | actual_date_completion_repair | date_order | date_acceptance_for_repair | sheduled_repair_co
mpletion_date
------+-------------------------+-------------+---------------+-------------------------------+------------+----------------------------+------------------
--------------
 3797 |                      85 |           5 |           575 |                               | 2025-04-30 | 2025-04-30                 | 2025-04-30
(1 строка)
```

Результат выполнения:

```
car_workshop=# INSERT INTO service_provided (id_service, id_contract, employee_service_number, date_acceptance_for_repair, sheduled_repair_completion_date,
service_cost)
car_workshop-# VALUES (2, 3797, 1, '2025-04-30', '2025-04-30', 1200);
ОШИБКА:  The employee works at a car workshop at a different address
КОНТЕКСТ:  функция PL/pgSQL check_employee_workshop(), строка 10, оператор RAISE
```

3.3 Дополнительное задание

Модифицированная триггерная функция для проверки корректности входа и выхода сотрудника:

```
emp_time=# create or replace function fn_check_time_punch()
emp_time-# returns trigger as $$
emp_time$# declare
emp_time$#     last_punch_time timestamp;
emp_time$#     last_is_out_punch boolean;
emp_time$# begin
emp_time$#     select punch_time, is_out_punch
emp_time$#     into last_punch_time, last_is_out_punch
emp_time$#     from time_punch
emp_time$#     where employee_id = new.employee_id
emp_time$#     order by punch_time desc
emp_time$#     limit 1;
emp_time$#
emp_time$#     if last_punch_time is not null then
emp_time$#         if new.is_out_punch = last_is_out_punch then
emp_time$#             return null;
emp_time$#         end if;
emp_time$#
emp_time$#         if new.punch_time <= last_punch_time then
emp_time$#             return null;
emp_time$#         end if;
emp_time$#
emp_time$#     else
emp_time$#         if new.is_out_punch = true then
emp_time$#             return null;
emp_time$#         end if;
emp_time$#     end if;
emp_time$#
emp_time$#     return new;
emp_time$# end;
emp_time$# $$ language plpgsql;
CREATE FUNCTION
```

Результат работы триггера:

```
emp_time=# insert into time_punch values (1, 1, true, '2025-01-01 10:00:00', 3);
INSERT 0 0
emp_time=# insert into time_punch values (1, 1, false, '2025-01-01 10:00:00', 3);
INSERT 0 1
emp_time=# insert into time_punch values (2, 1, false, '2025-01-01 11:00:00', 3);
INSERT 0 0
emp_time=# insert into time_punch values (2, 1, true, '2025-01-01 10:00:00', 3);
INSERT 0 0
emp_time=# insert into time_punch values (2, 1, true, '2025-01-01 11:00:00', 3);
INSERT 0 1
emp_time=# insert into time_punch values (3, 1, false, '2024-01-01 09:00:00', 3);
INSERT 0 0
```

```
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |     punch_time      | change_employee_id
----+-------------+--------------+---------------------+--------------------
  1 |           1 | f            | 2025-01-01 10:00:00 |                  3
  2 |           1 | t            | 2025-01-01 11:00:00 |                  3
(2 строки)
```

Что делает триггер:

– Запрещает два одинаковых действия подряд;

– Проверяет, что время новой отметки строго позже последней (сравнивается и дата, и время);

– Проверяет, что первая запись для сотрудника обязательно должна быть входом.

4 Вывод

В ходе выполнения лабораторной работы были получены практические навыки создания и использования хранимых процедур и функций, триггеров в СУБД PostgreSQL. Были разработаны три процедуры, соответствующие индивидуальному варианту задания, что позволило автоматизировать выполнение определённых операций с базой данных. Кроме того, было создано семь оригинальных триггеров, обеспечивающих реакцию на различные события в базе данных, такие как добавление и обновление записей.

Работа показала важность использования триггеров для обеспечения целостности данных, автоматизации бизнес-логики и упрощения взаимодействия с базой данных.