Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе 3 «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

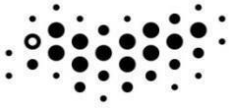
по дисциплине «**Проектирование и реализация баз данных**»

Автор: Никифорова Анна Дмитриевна

Факультет: Инфокоммуникационные технологии

Группа: K32421

Преподаватель: Говорова Марина Михайловна



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

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

Практическое задание:

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).

2. 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.

3. Создать авторский триггер по варианту индивидуального задания.

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

Выполнение:

Индивидуальное практическое задание: предметная область «Школа».

Создать хранимые процедуры:

- для расчета общей нагрузки преподавателя за определенную дату;
- для смены старосты класса (должна быть проверка, что новый староста из того же класса).
- для расформирования класса.

Создать триггер:

- нельзя изменить факт посещения учеником урока на true, если ученик в дату проведения урока не учился в школе (то есть был отчислен или находился в академ. отпуске).

1.

```
CREATE OR REPLACE FUNCTION get_teacher_overall_hours(teacher_personnel_number_
int, date_ date) RETURNS int LANGUAGE PLPGSQL AS $$
DECLARE
	res int;
BEGIN
	res =
    (SELECT SUM(hours)
    FROM "School_schema"."Workload"
    WHERE teacher_personnel_number = teacher_personnel_number_
        AND date_from <= date_
```

```
        AND date_to >= date_
    GROUP BY teacher_personnel_number);

      IF res IS NULL THEN
            RETURN 0;
      END IF;
      RETURN res;
END;
$$;
```

```
School=# select * from get_teacher_overall_hours(1, '2022-01-01');
 get_teacher_overall_hours
---------------------------
                       360
(1 строка)


School=# select * from get_teacher_overall_hours(11, '2025-01-01');
 get_teacher_overall_hours
---------------------------
                         0
(1 строка)
```

Рис. 1 – «Иллюстрация работы первой функции»

2.

```
CREATE OR REPLACE PROCEDURE change_headstudent(student_code_ int, class_name_
varchar) LANGUAGE PLPGSQL AS $$
BEGIN
      -- Временная таблица с информацией о том, где на текущий момент учится
переданный нам ученик
      CREATE TEMP TABLE this_student_in_some_class AS
                  (SELECT student_in_class_id,
                              class_id
                   FROM "School_schema"."Student_in_class"
                   WHERE student_code = student_code_
                              AND status LIKE 'Числится'
                              AND date_to >= CURRENT_DATE);

      -- Временная таблица с информацией об id того ныне существующего класса,
название которого нам передали
      CREATE TEMP TABLE this_class_id AS
                  (SELECT class_id
                   FROM "School_schema"."Class"
                   JOIN "School_schema"."Curriculum" USING(curriculum_id)
                   WHERE status LIKE 'Действует'
                              AND concat(class_number::varchar, class_letter) =
class_name_);

      -- Если id найденного ученика и класса совпали, то
      IF (
            (SELECT class_id
```

```
            FROM this_student_in_some_class) =
            (SELECT class_id
            FROM this_class_id)) THEN

            -- Временная таблица с информацией о текущем старосте данного
класса
            CREATE TEMP TABLE this_class_current_headstudent_id AS
                        (SELECT headstudent_id
                        FROM "School_schema"."Class"
                        JOIN "School_schema"."Student_in_class" USING(class_id)
                        JOIN "School_schema"."Headstudent"
USING(student_in_class_id)
                        WHERE "School_schema"."Headstudent".date_from <
CURRENT_DATE
                                    AND ("School_schema"."Headstudent".date_to
> CURRENT_DATE
                                                OR
"School_schema"."Headstudent".date_to IS NULL)
                                    AND class_id = (SELECT class_id FROM
this_class_id));

            -- Если у текущего старосты не закончился срок, заканчиваем его
            IF EXISTS(SELECT * FROM this_class_current_headstudent_id) THEN
                  UPDATE "School_schema"."Headstudent"
                  SET date_to = CURRENT_DATE
                  WHERE headstudent_id =
                                          (SELECT headstudent_id
                                          FROM
this_class_current_headstudent_id);
            END IF;

            -- Добавляем в таблицу информацию о новом старосте
            INSERT INTO "School_schema"."Headstudent" (student_in_class_id,

date_from)
            VALUES ((SELECT student_in_class_id FROM
this_student_in_some_class), CURRENT_DATE);
      END IF;

      -- Удаляем временные таблицы
      DROP TABLE IF EXISTS this_student_in_some_class;
      DROP TABLE IF EXISTS this_class_id;
      DROP TABLE IF EXISTS this_class_current_headstudent_id;
END;
$$;
```

```
School=# select concat(class_number::varchar, class_letter) as class_name, class_id from "School_schema"."Class"
School-# join "School_schema"."Curriculum" using(curriculum_id) where status = 'Действует';
 class_name | class_id
------------+----------
 5В         |       79
 5А         |       77
 6В         |       82
 6Б         |       81
 6А         |       80
 7В         |       85
 7Б         |       84
 7А         |       83
 8В         |       88
 8Б         |       87
 8А         |       86
 9В         |       91
 9Б         |       90
 9А         |       89
 10Б        |       93
 10А        |       92
 11Б        |       95
 11А        |       94
(18 строк)
```

Рис. 2 – «Действующие классы»

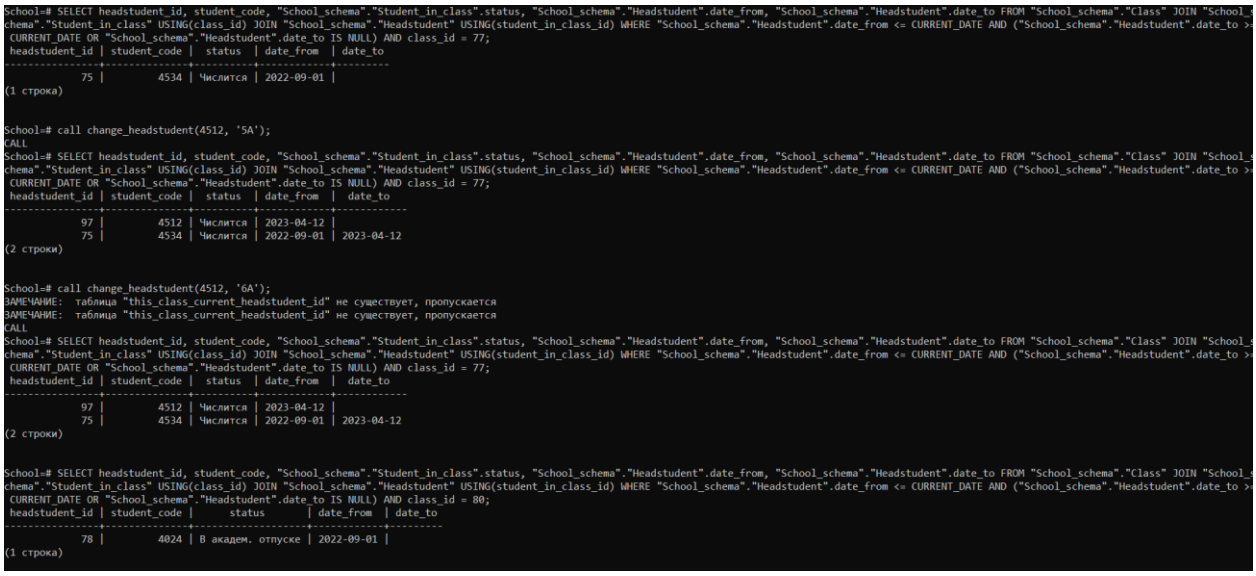


Рис. 3 – «Иллюстрация работы второй процедуры»

3.

```
CREATE OR REPLACE PROCEDURE class_no_longer_exists(class_name_ varchar) LANGUAGE
PLPGSQL AS $$
DECLARE
	class_id_ int;
BEGIN

	-- Найти ныне существующий класс по его названию
	class_id_ =
    (SELECT class_id
    FROM "School_schema"."Class"
    JOIN "School_schema"."Curriculum" USING(curriculum_id)
    WHERE status LIKE 'Действует'
	  AND concat(class_number::varchar, class_letter) = class_name_);
```

```
        -- Всем ученикам этого класса, которые не отчислены, поставить статус
'Числился'
        UPDATE "School_schema"."Student_in_class"
        SET status = 'Числился',
              date_to = CURRENT_DATE
        WHERE class_id = class_id_
                    AND status != 'Отчислен';

        -- Поставить классу статус 'Расформирован'
        UPDATE "School_schema"."Class"
        SET status = 'Расформирован'
        WHERE class_id = class_id_;
END;
$$;
```

```
School=# select concat(class_number::varchar, class_letter) as class_name, class_id from "School_schema"."Class"
School-# join "School_schema"."Curriculum" using(curriculum_id) where status = 'Действует' limit 3;
 class_name | class_id
------------+----------
 5B         |       79
 5A         |       77
 6B         |       82
(3 строки)


School=# select * from "School_schema"."Student_in_class" where class_id = 79 order by student_in_class_id limit 10;
 student_in_class_id | date_from  |  date_to   |      status       | class_id | student_code
---------------------+------------+------------+-------------------+----------+--------------
                2446 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4726
                2451 | 2022-09-01 | 2023-08-30 | Отчислен          |       79 |         4505
                2452 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4506
                2453 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4507
                2456 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4510
                2459 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4513
                2461 | 2022-09-01 | 2023-08-30 | В академ. отпуске |       79 |         4515
                2464 | 2022-09-01 | 2023-08-30 | Отчислен          |       79 |         4518
                2465 | 2022-09-01 | 2023-08-30 | Числится          |       79 |         4519
                2466 | 2022-09-01 | 2023-08-30 | В академ. отпуске |       79 |         4520
(10 строк)


School=# call class_no_longer_exists('5B');
CALL
School=# select status from "School_schema"."Class" where class_id = 79;
    status
---------------
 Расформирован
(1 строка)


School=# select * from "School_schema"."Student_in_class" where class_id = 79 order by student_in_class_id limit 10;
 student_in_class_id | date_from  |  date_to   |  status  | class_id | student_code
---------------------+------------+------------+----------+----------+--------------
                2446 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4726
                2451 | 2022-09-01 | 2023-08-30 | Отчислен |       79 |         4505
                2452 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4506
                2453 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4507
                2456 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4510
                2459 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4513
                2461 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4515
                2464 | 2022-09-01 | 2023-08-30 | Отчислен |       79 |         4518
                2465 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4519
                2466 | 2022-09-01 | 2023-04-12 | Числился |       79 |         4520
(10 строк)
```

Рис. 4 – «Иллюстрация работы третьей процедуры»

Триггер.

```
CREATE OR REPLACE FUNCTION fn_change_attendance() RETURNS TRIGGER AS $psql$
 BEGIN
        IF (NEW.attendance != OLD.attendance
```

```
            AND NEW.attendance
            AND EXISTS
                        (SELECT *
                        FROM "School_schema"."Attendance"
                        JOIN "School_schema"."Student_in_class"
                        USING(student_in_class_id)
                        JOIN "School_schema"."Lesson" USING(lesson_id)
                        WHERE attendance_id = NEW.attendance_id
                                    AND student_in_class_id =
                                    NEW.student_in_class_id
                                    AND date_from <= lesson_date
                                    AND lesson_date <= date_to
                                    AND status IN ('Числился', 'Числится'))) THEN
            RETURN NEW;
      END IF;
      RETURN (NEW.attendance_id,
                  FALSE,
                  NEW.note,
                  NEW.student_in_class_id,
                  NEW.lesson_id);
 END;
$psql$ LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER change_attendance
BEFORE
UPDATE ON "School_schema"."Attendance"
FOR EACH ROW EXECUTE PROCEDURE fn_change_attendance();
```

```
School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE date_from <= lesson_date AND lesson_date <= date_to
School-# AND status NOT IN ('Числился', 'Числится') AND NOT attendance LIMIT 5;
 attendance_id | lesson_date | attendance | note | student_in_class_id | date_from  |  date_to   |  status
---------------+-------------+------------+------+---------------------+------------+------------+----------
            42 | 2018-09-12  | f          |      |                  19 | 2018-09-01 | 2019-08-30 | Отчислен
            52 | 2018-09-12  | f          |      |                  56 | 2018-09-01 | 2019-08-30 | Отчислен
            70 | 2018-09-19  | f          |      |                   4 | 2018-09-01 | 2019-08-30 | Отчислен
           184 | 2018-10-10  | f          |      |                  56 | 2018-09-01 | 2019-08-30 | Отчислен
             9 | 2018-09-05  | f          |      |                  19 | 2018-09-01 | 2019-08-30 | Отчислен
(5 строк)


School=# UPDATE "School_schema"."Attendance" SET attendance = true, note = 'Changed?' WHERE attendance_id = 42;
UPDATE 1
School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE attendance_id = 42;
 attendance_id | lesson_date | attendance |   note   | student_in_class_id | date_from  |  date_to   |  status
---------------+-------------+------------+----------+---------------------+------------+------------+----------
            42 | 2018-09-12  | f          | Changed? |                  19 | 2018-09-01 | 2019-08-30 | Отчислен
(1 строка)


School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE date_from <= lesson_date AND lesson_date <= date_to
School-# AND status IN ('Числился', 'Числится') AND attendance LIMIT 5;
 attendance_id | lesson_date | attendance | note | student_in_class_id | date_from  |  date_to   |  status
---------------+-------------+------------+------+---------------------+------------+------------+----------
             3 | 2018-09-05  | t          |      |                   3 | 2018-09-01 | 2019-08-30 | Числился
             5 | 2018-09-05  | t          |      |                   7 | 2018-09-01 | 2019-08-30 | Числился
             6 | 2018-09-05  | t          |      |                   8 | 2018-09-01 | 2019-08-30 | Числился
            10 | 2018-09-05  | t          |      |                  23 | 2018-09-01 | 2019-08-30 | Числился
            12 | 2018-09-05  | t          |      |                  28 | 2018-09-01 | 2019-08-30 | Числился
(5 строк)


School=# UPDATE "School_schema"."Attendance" SET attendance = false, note = 'Changed?' WHERE attendance_id = 3;
UPDATE 1
School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE attendance_id = 3;
 attendance_id | lesson_date | attendance |   note   | student_in_class_id | date_from  |  date_to   |  status
---------------+-------------+------------+----------+---------------------+------------+------------+----------
             3 | 2018-09-05  | f          | Changed? |                   3 | 2018-09-01 | 2019-08-30 | Числился
(1 строка)
```

Рис. 5 – «Иллюстрация работы триггера»

```
School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE date_from <= lesson_date AND lesson_date <= date_to
School-# AND status IN ('Числился', 'Числится') AND not attendance LIMIT 5;
 attendance_id | lesson_date | attendance | note | student_in_class_id | date_from  |  date_to   |  status
---------------+-------------+------------+------+---------------------+------------+------------+----------
            11 | 2018-09-05  | f          |      |                  27 | 2018-09-01 | 2019-08-30 | Числился
            13 | 2018-09-05  | f          |      |                  30 | 2018-09-01 | 2019-08-30 | Числился
            14 | 2018-09-05  | f          |      |                  33 | 2018-09-01 | 2019-08-30 | Числился
            18 | 2018-09-05  | f          |      |                  55 | 2018-09-01 | 2019-08-30 | Числился
            20 | 2018-09-05  | f          |      |                  58 | 2018-09-01 | 2019-08-30 | Числился
(5 строк)


School=# UPDATE "School_schema"."Attendance" SET attendance = true, note = 'Changed?' WHERE attendance_id = 13;
UPDATE 1
School=# SELECT attendance_id, lesson_date, attendance, note, student_in_class_id, date_from, date_to, status
School-# FROM "School_schema"."Attendance" JOIN "School_schema"."Student_in_class" USING(student_in_class_id)
School-# JOIN "School_schema"."Lesson" USING(lesson_id) WHERE attendance_id = 13;


            13 | 2018-09-05  | t          | Changed? |                  30 | 2018-09-01 | 2019-08-30 | Числился
```

Рис. 6 – «Иллюстрация работы триггера»

Триггер из практической работы.

Логика работы: для поступившей на вход записи из таблицы выбирается самая последняя по времени запись для того же сотрудника. Если новая запись – вход, то выбранная запись должна либо отсутствовать, либо быть выходом. Иначе выбранная запись должна быть входом. Также время добавляемой записи должно быть больше времени выбранной записи и быть меньше или равно текущему времени.

```
CREATE OR REPLACE FUNCTION fn_check_time_punch() RETURNS TRIGGER AS $psql$
      BEGIN
            IF (NOT exists
                  (SELECT *
                  FROM time_punch
                  WHERE employee_id = new.employee_id
                  ORDER BY punch_time DESC
                  LIMIT 1)
                AND new.punch_time <= now()
                AND NOT new.is_out_punch)
              OR (
                  (SELECT punch_time
                  FROM
                      (SELECT *
                      FROM time_punch
                      WHERE employee_id = new.employee_id
                      ORDER BY punch_time DESC
                      LIMIT 1) AS temp_table) < new.punch_time
                AND new.punch_time <= now()
                AND
                    (SELECT is_out_punch
                    FROM
                        (SELECT *
                        FROM time_punch
                        WHERE employee_id = new.employee_id
                        ORDER BY punch_time DESC
                        LIMIT 1) AS temp_table) != new.is_out_punch) THEN
              RETURN NEW;
            END IF;
            RETURN NULL;
      END;
$psql$ LANGUAGE PLPGSQL;

DROP TRIGGER IF EXISTS check_time_punch ON time_punch;

CREATE TRIGGER check_time_punch BEFORE INSERT ON time_punch
FOR EACH ROW EXECUTE PROCEDURE fn_check_time_punch();
```

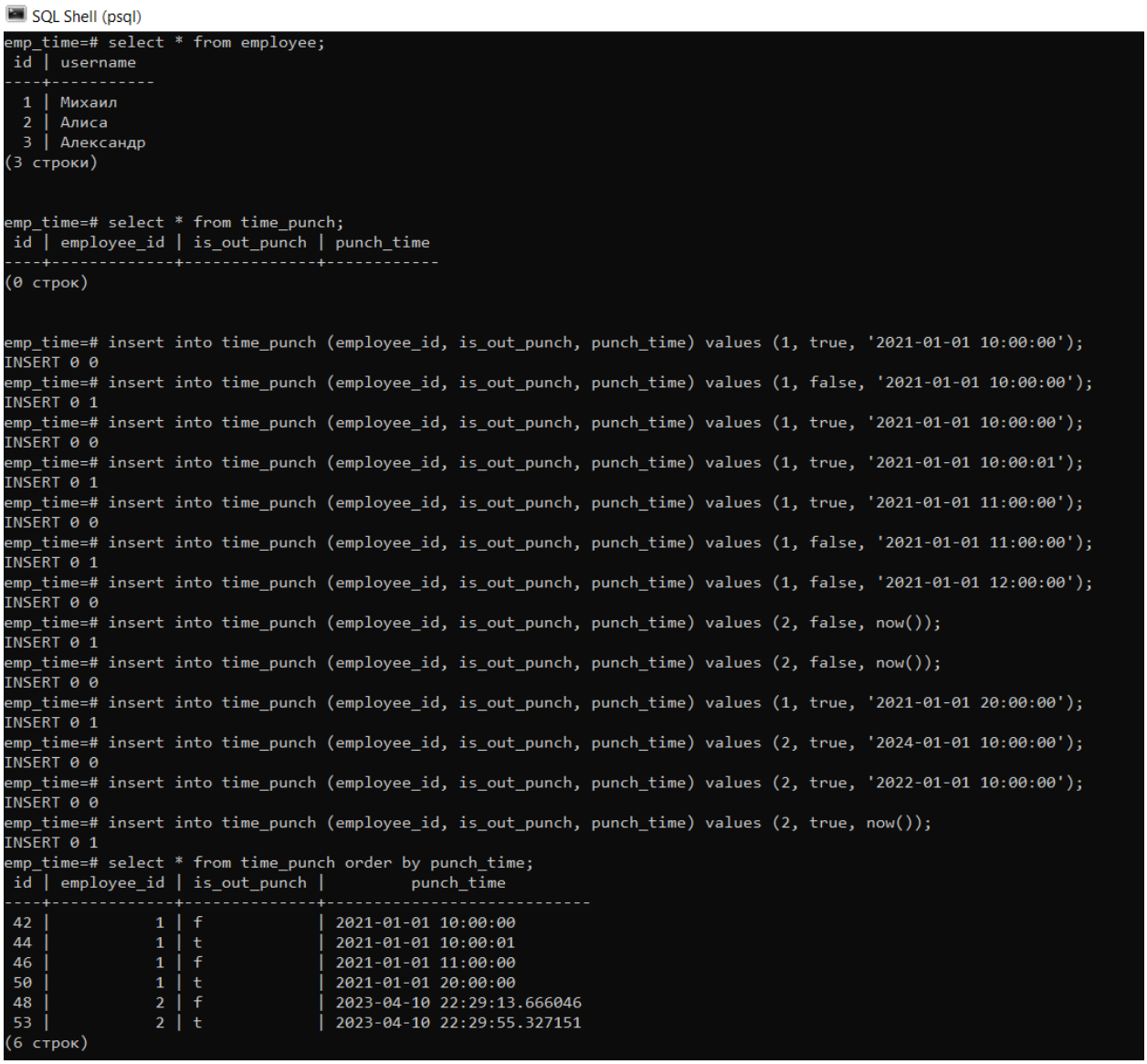


Рис. 7 – «Иллюстрация работы переписанного триггера»

Выводы:

В рамках данной лабораторной работы я приобрела практические навыки создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Также я научилась объявлять переменные и создавать временные таблицы внутри функций. Оказалось, что если в функции создается временная таблица, то нужно не забыть её удалить, иначе при последующих вызовах функции будет использоваться первая таблица (без учета возможных модификаций данных).

На мой взгляд, возможность создания переменных и временных таблиц, с одной стороны, приближает SQL к широко используемым языкам программирования (Python, C++). С другой стороны, SQL всё еще не хватает гибкости – неудобная инициализация, неочевидный синтаксис, нехватка

встроенных функций. Код не получается тестировать, так как всё необходимо оборачивать в SELECT-ы, чтобы увидеть результат. По большей части ты пишешь код вслепую и надеешься, что он не упадет. Нет возможности откатить какое-либо действие и вернуться к предыдущему состоянию. Также, очень сложно найти адекватное и рабочее решение на StackOverfow. В целом, мне кажется, работать с табличными данными через чистый SQL крайне неудобно. Гораздо удобнее использовать, например, библиотеку pandas в Python.

Триггеры же – прекрасная вещь. Особенно мне понравилась возможность использовать триггеры для логирования.