

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: Пиотуховский А.А.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание	3
Вариант 10. БД «Автовокзал»	3
Схема базы данных	4
Выполнение	6
Хранимая процедура продажи билета.....	6
Хранимая процедура возврата билета.....	8
Хранимая процедура добавления нового рейса	9
Доработка триггера с практики.....	10
Авторский триггер	13
Вывод.....	14

Цель работы

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

Практическое задание

Вариант 2

1. Создать процедуры/функции 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.

Вариант 10. БД «Автовокзал»

Описание предметной области:

С автовокзала ежедневно отправляется несколько междугородных/международных автобусных рейсов. Номер рейса определяется маршрутом и временем отправления. По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

Автобусы курсируют по расписанию, но могут назначаться дополнительные рейсы на заданный период или определенные даты.

Билеты могут продаваться предварительно, но не ранее чем за 10 суток. В билете указывается номер места в автобусе. На каждый рейс может продаваться не более 10 билетов без места, цена на которые снижается на 10%. Пунктами отправления и назначения, согласно билету, могут быть промежуточные остановки.

Билеты могут продаваться в кассе автовокзала или онлайн.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные остановки согласно купленному билету.

На каждый рейс формируется экипаж из двух водителей.

БД должна содержать следующий минимальный набор сведений: Номер рейса. Номер водителя. Номер автобуса. Паспортные данные водителя. Пункт отправления. Пункт назначения. Промежуточные остановки. Дата отправления. Время отправления. Время в пути. Тип автобуса. Количество мест в автобусе. Страна. Производитель. Год выпуска. Номер билета. Номер места в автобусе (при наличии). Цена билета. ФИО пассажира. Паспортные данные пассажира.

Схема базы данных

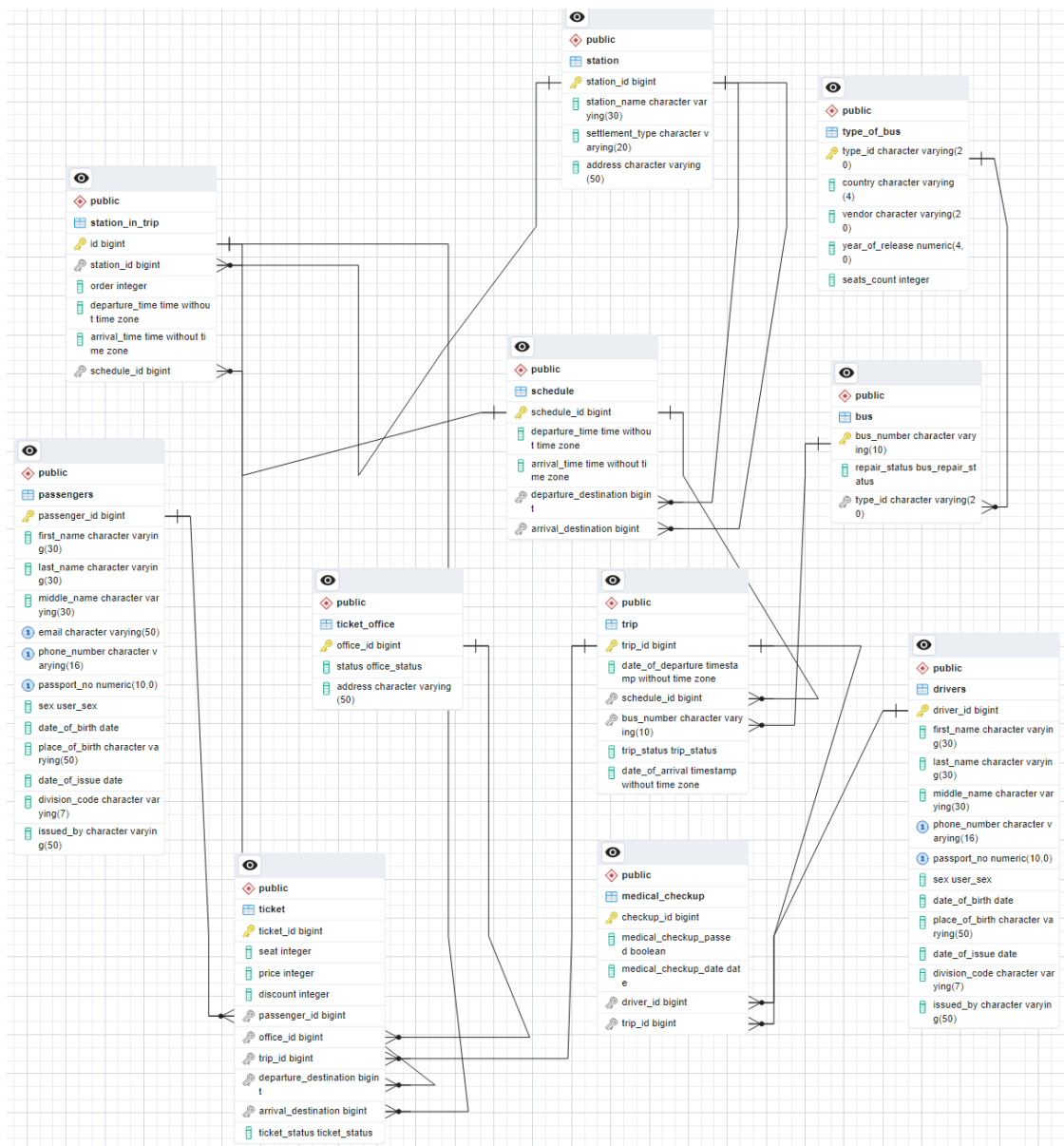


Рисунок 1 – Схема логической модели базы данных в pgAdmin4.

ERD в pgAdmin 4 строит верные, но запутанные связи. Без пересоздания всех связей собрать таблицы в аккуратном порядке не получилось. На рисунке 2 представлена эта же логическая схема, но выполненная в IDEA от JetBrains (Powered by yFiles).

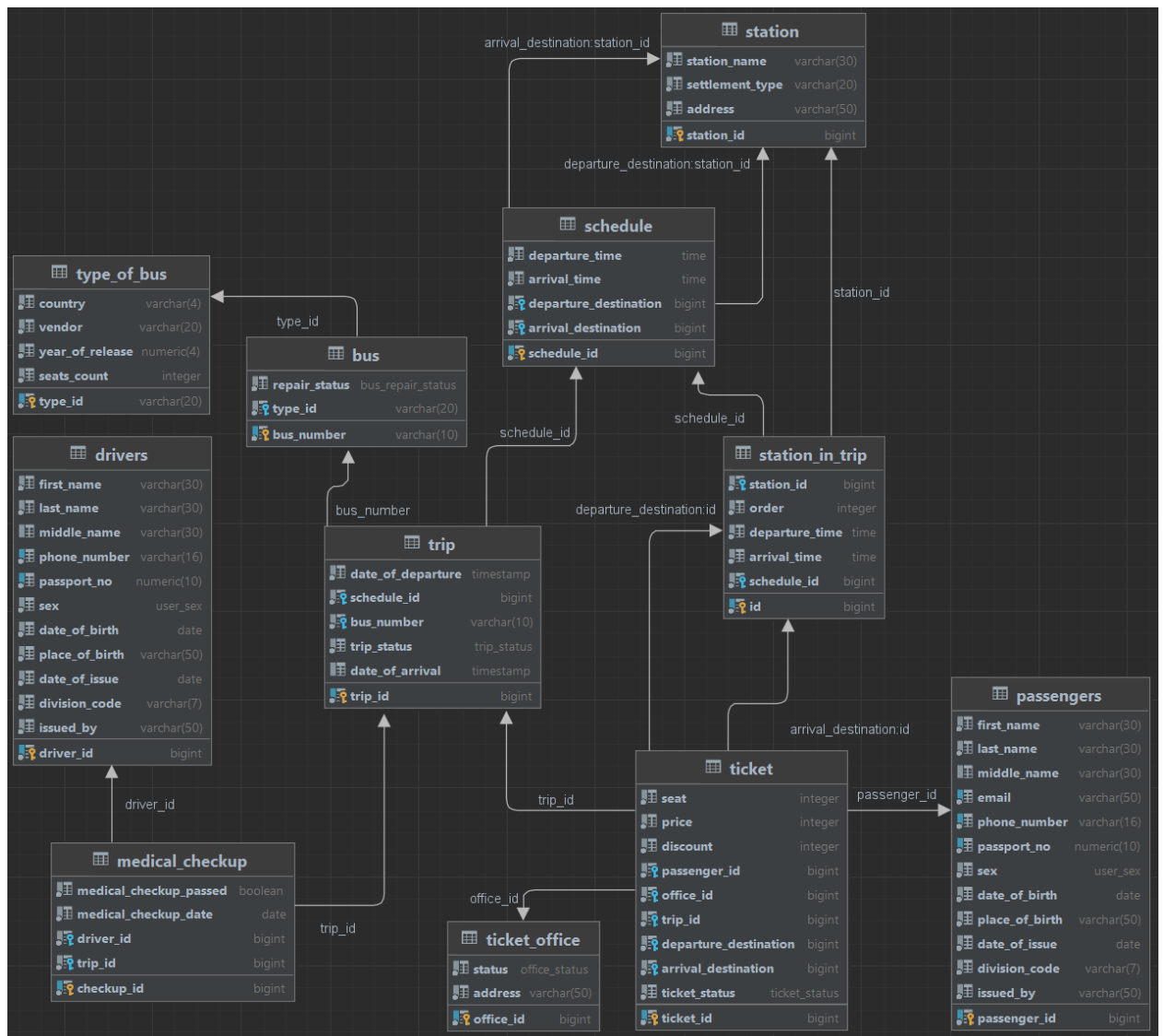


Рисунок 2 – Схема логической модели базы данных в IDEA от JetBrains.

Выполнение

Хранимая процедура продажи билета

```
CREATE OR REPLACE FUNCTION sell_ticket(
    IN _seat INTEGER,
    IN _price INTEGER,
    IN _passenger_id BIGINT,
    IN _office_id BIGINT,
    IN _trip_id BIGINT,
    IN _departure_destination BIGINT,
    IN _arrival_destination BIGINT,
    IN _discount INTEGER DEFAULT 0
)
    RETURNS bigint
    LANGUAGE plpgsql
AS
$$
DECLARE
    _ticket_id INT;
BEGIN
    IF _seat IS NULL THEN
        RAISE EXCEPTION 'Не передано значение seat';
    END IF;
    IF _price IS NULL THEN
        RAISE EXCEPTION 'Не передано значение price';
    END IF;
    IF _passenger_id IS NULL THEN
        RAISE EXCEPTION 'Не передано значение passenger_id';
    END IF;
    IF _office_id IS NULL THEN
        RAISE EXCEPTION 'Не передано значение office_id';
    END IF;
    IF _trip_id IS NULL THEN
        RAISE EXCEPTION 'Не передано значение trip_id';
    END IF;
    IF _departure_destination IS NULL THEN
        RAISE EXCEPTION 'Не передано значение departure_destination';
    END IF;
    IF _arrival_destination IS NULL THEN
        RAISE EXCEPTION 'Не передано значение arrival_destination';
    END IF;

    IF NOT EXISTS (SELECT 1
                   FROM passengers
                   WHERE passenger_id = _passenger_id) THEN
        RAISE EXCEPTION 'Пассажир с таким id не существует';
    END IF;

    IF NOT EXISTS (SELECT 1
                   FROM ticket_office
                   WHERE office_id = _office_id
                     and status = 'opened'::office_status) THEN
        RAISE EXCEPTION 'Касса с таким id не существует';
    END IF;

    IF NOT EXISTS (SELECT 1
                   FROM trip
                   WHERE trip_id = _trip_id) THEN
        RAISE EXCEPTION 'Рейс с таким id не существует';
    END IF;
```

```

END IF;

IF NOT EXISTS (SELECT 1
                FROM station_in_trip
                WHERE station_id = _departure_destination) THEN
    RAISE EXCEPTION 'Станция отправления с таким id не существует';
END IF;

IF NOT EXISTS (SELECT 1
                FROM station_in_trip
                WHERE station_id = _arrival_destination) THEN
    RAISE EXCEPTION 'Станция назначения с таким id не существует';
END IF;

INSERT INTO ticket (seat,
                    price,
                    discount,
                    passenger_id,
                    office_id,
                    trip_id,
                    departure_destination,
                    arrival_destination)

VALUES (_seat,
        _price,
        _discount,
        _passenger_id,
        _office_id,
        _trip_id,
        _departure_destination,
        _arrival_destination)
returning ticket_id into _ticket_id;
RETURN _ticket_id;
END;
$$;

```

Пример выполнения

```
SELECT sell_ticket(5, 8, 13, 2, 3, 2, 3);
```

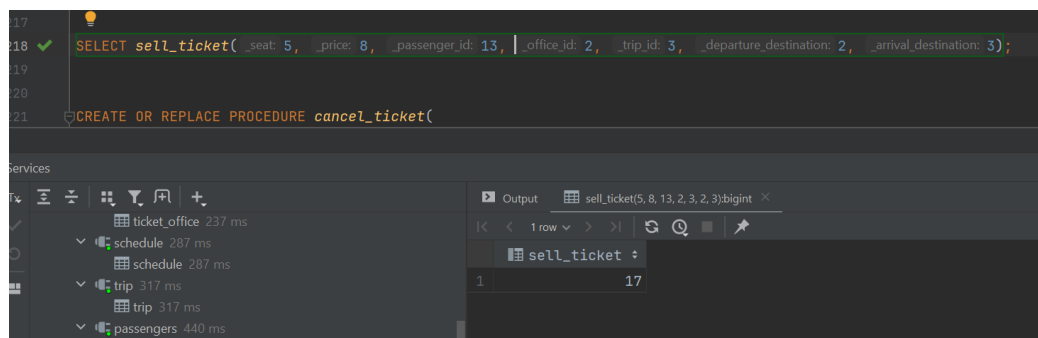
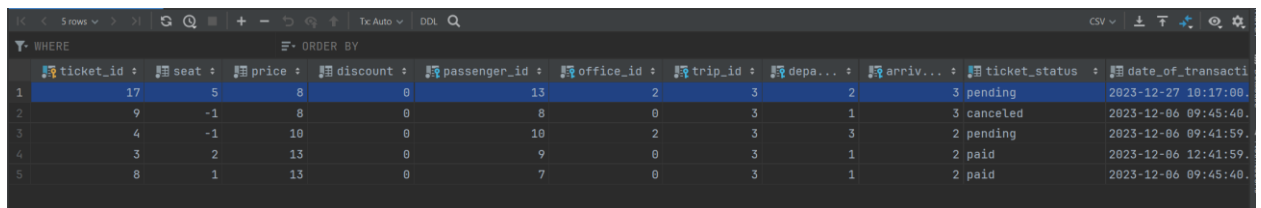


Рисунок 3 – Результат работы процедуры по продаже билета

Хранимая процедура возврата билета

```
CREATE OR REPLACE PROCEDURE cancel_ticket(  
    IN _ticket_id BIGINT  
)  
    LANGUAGE plpgsql  
AS  
$$  
BEGIN  
    IF NOT EXISTS (SELECT 1  
                   FROM ticket  
                   WHERE ticket_id = _ticket_id) THEN  
        RAISE EXCEPTION 'Билет с таким id не существует';  
    END IF;  
  
    IF (SELECT ticket_status  
        FROM ticket  
        WHERE ticket_id = _ticket_id) = 'canceled'::ticket_status THEN  
        RAISE EXCEPTION 'Билет с таким id уже отменен';  
    END IF;  
  
    UPDATE ticket  
    SET ticket_status = 'canceled'::ticket_status  
    WHERE ticket_id = _ticket_id;  
END;  
$$;
```

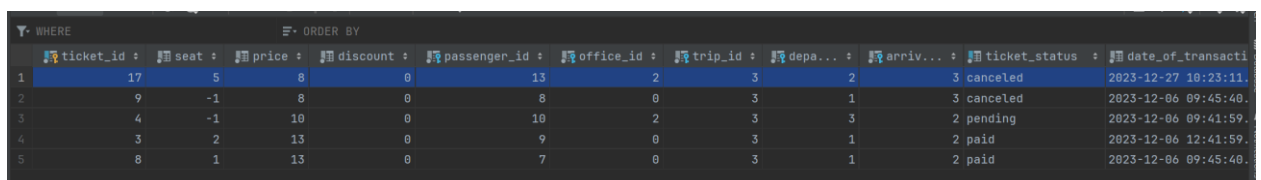
Пример выполнения



	ticket_id	seat	price	discount	passenger_id	office_id	trip_id	depa...	arriv...	ticket_status	date_of_transacti
1	17	5	8	0	13	2	3	2	3	pending	2023-12-27 10:17:00.
2	9	-1	8	0	8	0	3	1	3	canceled	2023-12-06 09:45:40.
3	4	-1	10	0	10	2	3	3	2	pending	2023-12-06 09:41:59.
4	3	2	13	0	9	0	3	1	2	paid	2023-12-06 12:41:59.
5	8	1	13	0	7	0	3	1	2	paid	2023-12-06 09:45:40.

Рисунок 4 – Вид таблицы до применения процедуры по отмене билета

```
CALL cancel_ticket(17);
```



	ticket_id	seat	price	discount	passenger_id	office_id	trip_id	depa...	arriv...	ticket_status	date_of_transacti
1	17	5	8	0	13	2	3	2	3	canceled	2023-12-27 10:23:11.
2	9	-1	8	0	8	0	3	1	3	canceled	2023-12-06 09:45:40.
3	4	-1	10	0	10	2	3	3	2	pending	2023-12-06 09:41:59.
4	3	2	13	0	9	0	3	1	2	paid	2023-12-06 12:41:59.
5	8	1	13	0	7	0	3	1	2	paid	2023-12-06 09:45:40.

Рисунок 5 – Вид таблицы после применения процедуры по отмене билета

Хранимая процедура добавления нового рейса

```
CREATE OR REPLACE FUNCTION create_trip(
    IN _schedule_id BIGINT,
    IN _bus_number VARCHAR(10),
    IN _date_of_departure TIMESTAMP WITH TIME ZONE,
    IN _date_of_arrival TIMESTAMP WITH TIME ZONE
)
    RETURNS bigint
    LANGUAGE plpgsql
AS
$$
DECLARE
    _trip_id BIGINT;
BEGIN
    IF _schedule_id IS NULL THEN
        RAISE EXCEPTION 'Не передано значение schedule_id';
    END IF;
    IF _bus_number IS NULL THEN
        RAISE EXCEPTION 'Не передано значение bus_number';
    END IF;
    IF _date_of_departure IS NULL THEN
        RAISE EXCEPTION 'Не передано значение date_of_departure';
    END IF;
    IF _date_of_arrival IS NULL THEN
        RAISE EXCEPTION 'Не передано значение date_of_arrival';
    END IF;

    IF NOT EXISTS (SELECT 1
                    FROM bus
                    WHERE bus_number = _bus_number) THEN
        RAISE EXCEPTION 'Автобус с таким номером не существует';
    END IF;

    IF NOT EXISTS (SELECT 1
                    FROM schedule
                    WHERE schedule_id = _schedule_id) THEN
        RAISE EXCEPTION 'Маршрут с таким id не существует';
    END IF;

    IF EXISTS(SELECT 1
              FROM trip
              WHERE bus_number = _bus_number
                 AND (_date_of_departure BETWEEN date_of_arrival and
date_of_departure
                 OR _date_of_arrival BETWEEN date_of_arrival and
date_of_departure)
                 AND trip_status != 'canceled'::trip_status) THEN
        RAISE EXCEPTION 'Наложение расписания! Этот автобус уже используется
в рейсе в это время.';
    end if;

    INSERT INTO trip (date_of_departure, schedule_id, bus_number,
trip_status, date_of_arrival)
    VALUES (_date_of_departure,
            _schedule_id,
            _bus_number,
            'on schedule'::trip_status,
            _date_of_arrival)
    returning trip_id into _trip_id;
    RETURN _trip_id;
```

```
END;  
$$;
```

Пример выполнения

```
SELECT create_trip(3, 'p141hh78', now() + INTERVAL '1 HOUR', now());
```

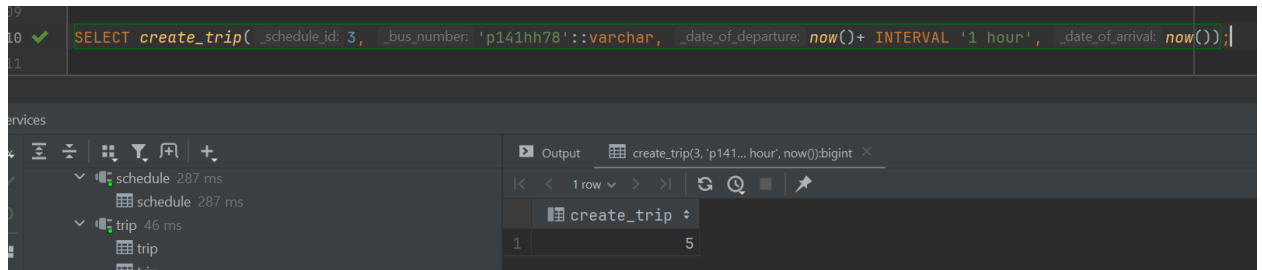


Рисунок 5 – Вид таблицы после применения процедуры по добавлению нового рейса

Доработка триггера с практики

```
create or replace function fn_check_time_punch() returns trigger as  
$psql$  
begin  
    if new.is_out_punch = (select tps.is_out_punch  
                           from time_punch tps  
                           where tps.employee_id = new.employee_id  
                           order by tps.id desc  
                           limit 1)  
        or (new.is_out_punch and not EXISTS(select tps.is_out_punch  
                                              from time_punch tps  
                                              where tps.employee_id =  
new.employee_id))  
        or new.punch_time > now()  
        or new.punch_time <= (select max(tps.punch_time)  
                              from time_punch tps  
                              where tps.employee_id = new.employee_id) then  
        return null;  
    end if;  
    return new;  
end;  
$psql$ language plpgsql;  
  
create trigger check_time_punch  
before insert  
on time_punch  
for each row  
execute procedure fn_check_time_punch();
```

console_3 [ifmo@localhost] x time_punch [ifmo@localhost] x

WHERE ORDER BY id

	id	employee_id	is_out_punch	punch_time
4	39	1	true	2023-11-24 14:03:43.655508
5	41	1	false	2023-11-24 14:03:49.725655
6	45	1	true	2023-11-24 14:04:24.365692
7	47	2	false	2023-11-24 14:04:38.444066
8	48	2	true	2023-11-24 14:04:50.927851
9	49	2	false	2023-11-24 14:38:44.691362
10	52	1	false	2023-11-24 14:38:55.269581
11	53	2	true	2023-11-24 14:39:01.750889
12	<generat...	1	false	<default>

Unexpected update count received (Actual: 0, Expected: 1). All changes will be rolled back.

Рисунок 6 – Невозможность два раза войти

console_3 [ifmo@localhost] x time_punch [ifmo@localhost] x

WHERE ORDER BY id

	id	employee_id	is_out_punch	punch_time
4	39	1	true	2023-11-24 14:03:43.655508
5	41	1	false	2023-11-24 14:03:49.725655
6	45	1	true	2023-11-24 14:04:24.365692
7	47	2	false	2023-11-24 14:04:38.444066
8	48	2	true	2023-11-24 14:04:50.927851
9	49	2	false	2023-11-24 14:38:44.691362
10	52	1	false	2023-11-24 14:38:55.269581
11	53	2	true	2023-11-24 14:39:01.750889
12	57	1	true	2023-12-27 10:57:42.705273
13	<generat...	1	true	<default>

Unexpected update count received (Actual: 0, Expected: 1). All changes will be rolled back.

Рисунок 7 – Невозможность два раза выйти

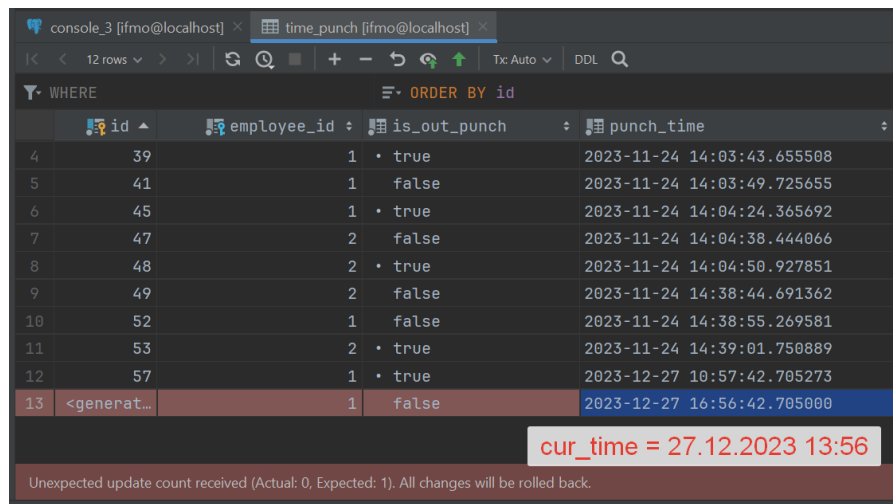
console_3 [ifmo@localhost] x time_punch [ifmo@localhost] x

WHERE ORDER BY id

	id	employee_id	is_out_punch	punch_time
4	39	1	true	2023-11-24 14:03:43.655508
5	41	1	false	2023-11-24 14:03:49.725655
6	45	1	true	2023-11-24 14:04:24.365692
7	47	2	false	2023-11-24 14:04:38.444066
8	48	2	true	2023-11-24 14:04:50.927851
9	49	2	false	2023-11-24 14:38:44.691362
10	52	1	false	2023-11-24 14:38:55.269581
11	53	2	true	2023-11-24 14:39:01.750889
12	57	1	true	2023-12-27 10:57:42.705273
13	<generat...	1	false	2023-12-27 10:56:42.705000

Unexpected update count received (Actual: 0, Expected: 1). All changes will be rolled back.

Рисунок 8 – Невозможность пройти контроль раньше последнего контроля



console_3 [ifmo@localhost] x time_punch [ifmo@localhost] x

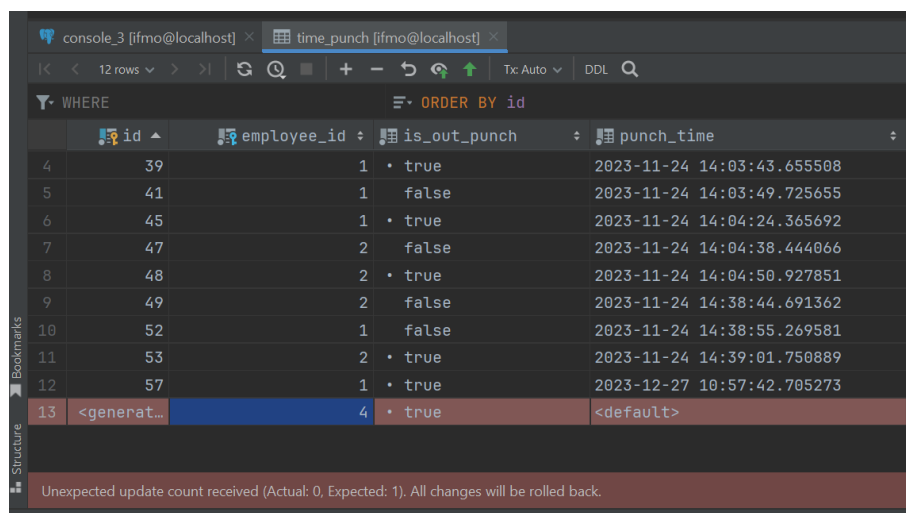
WHERE ORDER BY id

	id	employee_id	is_out_punch	punch_time
4	39	1	true	2023-11-24 14:03:43.655508
5	41	1	false	2023-11-24 14:03:49.725655
6	45	1	true	2023-11-24 14:04:24.365692
7	47	2	false	2023-11-24 14:04:38.444066
8	48	2	true	2023-11-24 14:04:50.927851
9	49	2	false	2023-11-24 14:38:44.691362
10	52	1	false	2023-11-24 14:38:55.269581
11	53	2	true	2023-11-24 14:39:01.750889
12	57	1	true	2023-12-27 10:57:42.705273
13	<generat...	1	false	2023-12-27 16:56:42.705000

cur_time = 27.12.2023 13:56

Unexpected update count received (Actual: 0, Expected: 1). All changes will be rolled back.

Рисунок 9 – Невозможность пройти контроль раньше, чем now()



console_3 [ifmo@localhost] x time_punch [ifmo@localhost] x

WHERE ORDER BY id

	id	employee_id	is_out_punch	punch_time
4	39	1	true	2023-11-24 14:03:43.655508
5	41	1	false	2023-11-24 14:03:49.725655
6	45	1	true	2023-11-24 14:04:24.365692
7	47	2	false	2023-11-24 14:04:38.444066
8	48	2	true	2023-11-24 14:04:50.927851
9	49	2	false	2023-11-24 14:38:44.691362
10	52	1	false	2023-11-24 14:38:55.269581
11	53	2	true	2023-11-24 14:39:01.750889
12	57	1	true	2023-12-27 10:57:42.705273
13	<generat...	4	true	<default>

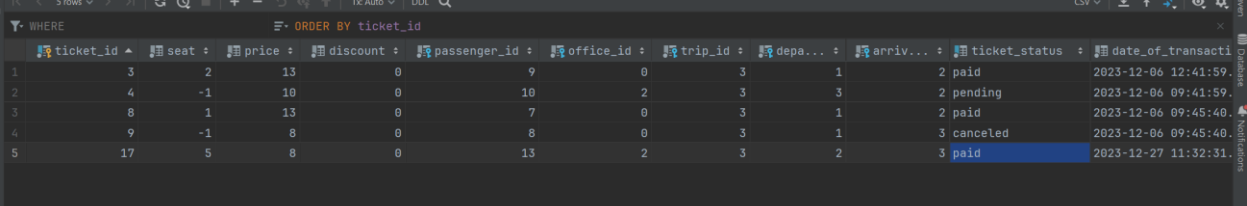
Unexpected update count received (Actual: 0, Expected: 1). All changes will be rolled back.

Рисунок 10 – Невозможность выйти, если ни разу не заходил

Авторский триггер

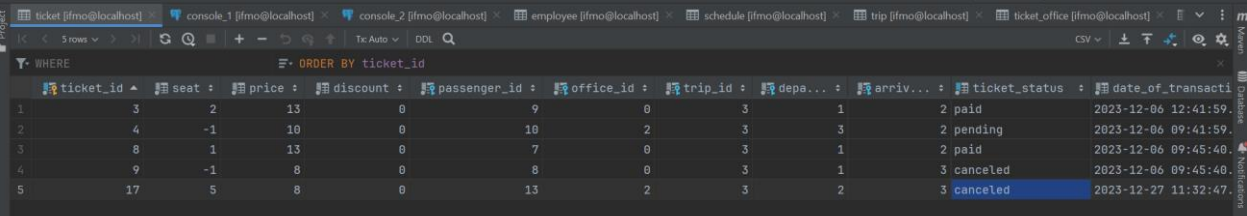
```
CREATE OR REPLACE FUNCTION update_date_of_transaction()
    RETURNS TRIGGER AS
$$
BEGIN
    NEW.date_of_transaction := NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_ticket_date
    BEFORE UPDATE
    ON ticket
    FOR EACH ROW
EXECUTE FUNCTION update_date_of_transaction();
```



	ticket_id	seat	price	discount	passenger_id	office_id	trip_id	depa...	arriv...	ticket_status	date_of_transacti
1	3	2	13	0	9	0	3	1	2	paid	2023-12-06 12:41:59.
2	4	-1	10	0	10	2	3	3	2	pending	2023-12-06 09:41:59.
3	8	1	13	0	7	0	3	1	2	paid	2023-12-06 09:45:40.
4	9	-1	8	0	8	0	3	1	3	canceled	2023-12-06 09:45:40.
5	17	5	8	0	13	2	3	2	3	paid	2023-12-27 11:32:31.

Рисунок 11 – вид данных до выполнения триггера



	ticket_id	seat	price	discount	passenger_id	office_id	trip_id	depa...	arriv...	ticket_status	date_of_transacti
1	3	2	13	0	9	0	3	1	2	paid	2023-12-06 12:41:59.
2	4	-1	10	0	10	2	3	3	2	pending	2023-12-06 09:41:59.
3	8	1	13	0	7	0	3	1	2	paid	2023-12-06 09:45:40.
4	9	-1	8	0	8	0	3	1	3	canceled	2023-12-06 09:45:40.
5	17	5	8	0	13	2	3	2	3	canceled	2023-12-27 11:32:47.

Рисунок 12 – вид данных после выполнения триггера

Вывод

В ходе выполнения лабораторной работы были ознакомлены с возможностями процедурного программирования в языке SQL, научились создавать процедуры, функции, а также использовать триггеры.