

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__» _____ 2025 г.

Проверено:

«__» _____ 2025 г.

Пользовательские функции и триггеры в PostgreSQL.

Отчёт по лабораторной работе №3

по дисциплине

«Управление данными»

Разработал студент гр. ИВТб-2301-05-00

_____/Черкасов А. А./
(подпись)

Старший Преподаватель

_____/Клюкин В. Л./
(подпись)

Работа защищена

«__» _____ 2025 г.

Киров
2025

Цели лабораторной работы

- познакомиться с созданием пользовательских функций и триггеров в PostgreSQL;
- освоить работу с составными типами данных и массивами;
- изучить основы работы с процедурным языком PL/pgSQL.

Задание

1. Для таблицы **devices** создать функцию **save_devices**, выполняющую вставку или обновление записи и возвращающую её **id**.
2. Для таблицы **hubs** (на которую ссылаются другие таблицы) создать функцию **delete_hubs**, проверяющую наличие внешних ссылок и выдающую ошибку при их наличии.
3. Для таблицы **devices** (содержащей числовой столбец **id**) создать функцию, возвращающую **SETOF** строк с **id** \geq заданного значения.
4. Создать составной тип **device_info** и функцию фильтрации массива объектов этого типа.
5. Создать таблицу лога **log_devices** и триггер для фиксации изменений имени устройства.
6. Реализовать функцию с динамически формируемым SQL-запросом.

Тема БД: «Система управления умным домом»

База данных предназначена для хранения информации о пользователях, хабах и устройствах умного дома, а также событий, которые генерируют эти устройства. Она обеспечивает:

- регистрацию и управление пользователями;
- хранение информации о хабах и их местоположении;
- классификацию устройств по типам и отслеживание их состояния;
- фиксацию событий устройств для мониторинга и анализа.

Реализация функций и триггеров

Все функции и триггеры реализованы на процедурном языке PL/pgSQL и описаны в файле `functions_triggers.sql`.

1. Функция `save_devices`

Функция принимает параметры устройства и выполняет вставку новой записи (если `_id IS NULL`) или обновление существующей. Возвращает идентификатор обработанной записи.

```
SELECT save_devices(NULL, 1, 4, 'New Smart Socket', '{"power": "off"}');  
SELECT save_devices(1, 1, 1, 'Main Ceiling Lamp', '{"power": "on", "brightness": 80}');
```

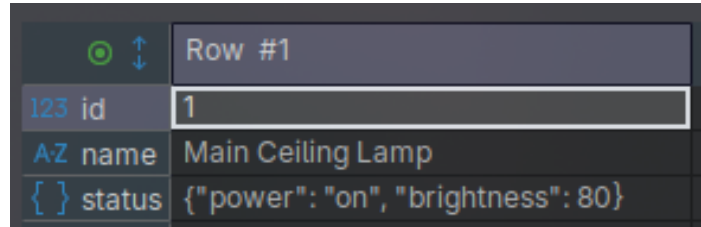


	Row #1
123 save_devices	5
123 save_devices	1

Рисунок 1.1 - Вывод функции `save_devices`.

Проверка:

```
SELECT id, name, status FROM devices WHERE id IN (1, 7);
```



	Row #1
123 id	1
A-Z name	Main Ceiling Lamp
{ } status	{"power": "on", "brightness": 80}

Рисунок 1.2 - Проверка функции save_devices.

2. Функция delete_hubs

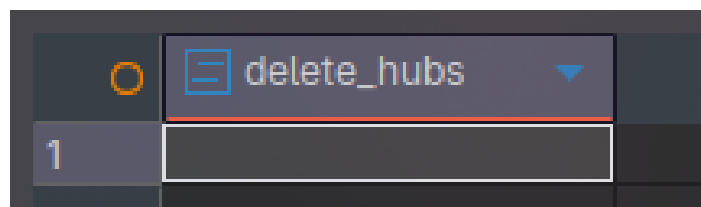
Перед удалением хаба проверяется наличие связанных устройств. При наличии внешних ссылок генерируется исключение:

«невозможно выполнить удаление, так как есть внешние ссылки.»

Для корректной работы ограничение внешнего ключа в таблице `devices` изменено на `ON DELETE RESTRICT`.

```
-- Попытка удалить хаб с привязанными устройствами (id=1)
SELECT delete_hubs(1);
-- Результат: ОШИБКА: невозможно выполнить удаление, так как есть внешние ссылки.

-- Удаление пустого хаба (если бы он был) - в нашем случае все хабы заняты,
-- но для демонстрации можно временно удалить устройства:
DELETE FROM devices WHERE hub_id = 4;
SELECT delete_hubs(4); -- Успешно
```



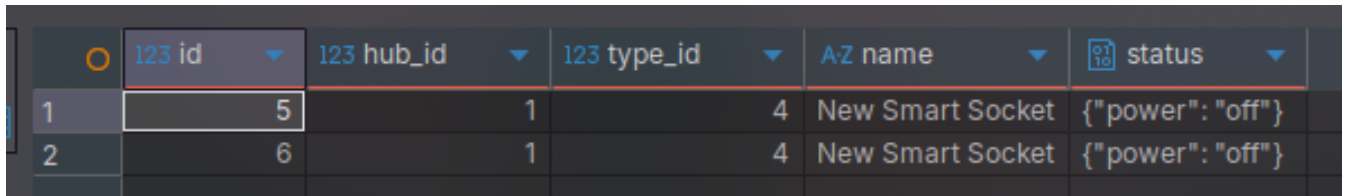
	delete_hubs	
1		

Рисунок 2 - Вывод функции delete_hubs – Успешно удалён.

3. Функция фильтрации filter_devices_by_id

Возвращает все устройства, у которых id больше или равен заданному значению.

```
SELECT * FROM filter_devices_by_id(5);
```



	123 id ▼	123 hub_id ▼	123 type_id ▼	A-Z name ▼	status ▼
1	5	1	4	New Smart Socket	{"power": "off"}
2	6	1	4	New Smart Socket	{"power": "off"}

Рисунок 3 - Вывод функции filter_device_by_id.

4. Составной тип и фильтрация массива

Создан составной тип:

```
create type device_info as (  
  id bigint,  
  name varchar(100),  
  type_name varchar(50),  
  hub_name varchar(100)  
);
```

Функция filter_device_array принимает массив объектов типа device_info и возвращает отфильтрованный массив.

```
SELECT filter_device_array(  
  ARRAY(  
    SELECT (d.id, d.name, dt.type_name, h.name)::device_info  
    FROM devices d  
    JOIN device_types dt ON d.type_id = dt.id  
    JOIN hubs h ON d.hub_id = h.id  
  ),  
  5  
);
```

filter_device_array				
	123 id	AZ name	AZ type_name	AZ hub_name
▶ 1	▶ 5	▶ New Smart Socket	▶ Smart Socket	▶ Living Room Hub

Рисунок 4 - Вывод функции filter_devices_array.

5. Таблица лога и триггер

Создана таблица `log_devices` для фиксации изменений имени устройств. Триггер `trigger_log_devices` срабатывает после операций `INSERT` и `UPDATE` и записывает старое и новое значение имени.

```
-- 5. таблица лога и триггер
create table if not exists log_devices (
  id bigserial primary key,
  device_id bigint references devices(id),
  change_time timestamp default now(),
  old_name varchar(100),
  new_name varchar(100)
);

create or replace function log_device_change()
returns trigger as $$
begin
  if tg_op = 'INSERT' then
    insert into log_devices (device_id, new_name)
    values (NEW.id, NEW.name);
  elsif tg_op = 'UPDATE' then
    insert into log_devices (device_id, old_name, new_name)
    values (NEW.id, OLD.name, NEW.name);
  end if;
  return NEW;
end;
$$ language plpgsql;

drop trigger if exists trigger_log_devices on devices;
```

```

create trigger trigger_log_devices
after insert or update on devices
for each row
execute function log_device_change();

```

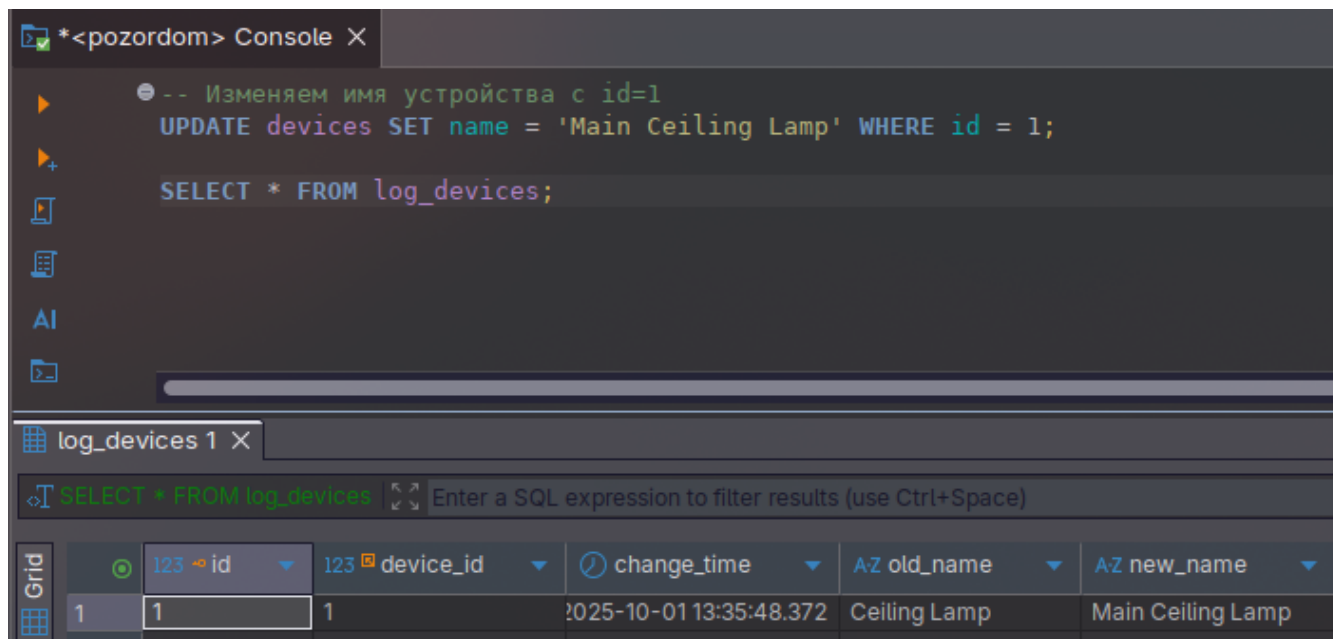


Рисунок 5 - Вывод функции log_devices.

6. Функция с динамическим SQL

Функция `get_column_value` принимает имя таблицы, имя столбца и идентификатор записи, формирует и выполняет динамический SQL-запрос, возвращая значение указанного столбца.

```

create or replace function get_column_value(
    table_name text,
    column_name text,
    record_id bigint
)
returns text as $$
declare
    result text;
begin

```

```

execute format(
    'select %I from %I where id = $1',
    column_name, table_name
)
into result
using record_id;
return result;
end;

```

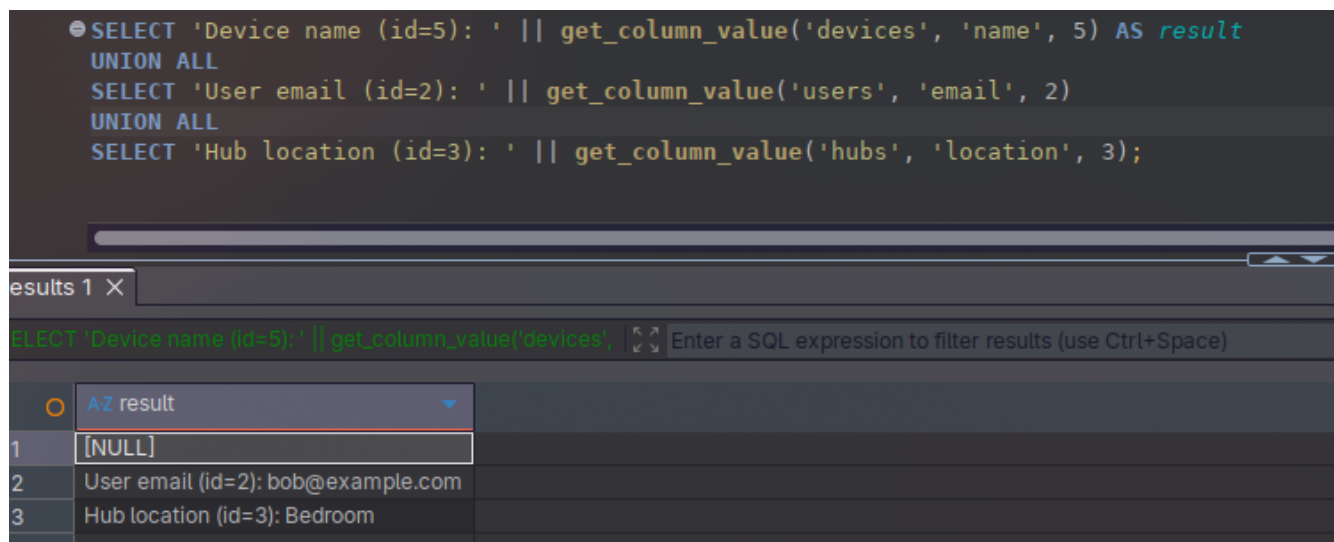


Рисунок 6 - Вывод функции `get_column_value`.

Вывод

В ходе выполнения лабораторной работы №3 были освоены механизмы создания пользовательских функций и триггеров в PostgreSQL. Реализованы функции для безопасной вставки, обновления и удаления записей с проверкой ссылочной целостности. Созданы составной тип и функция для работы с массивами. Реализовано логирование изменений с помощью триггера. Также освоено использование динамических SQL-запросов. Все функции написаны на процедурном языке PL/pgSQL и протестированы на корректность работы.

Приложение A1. Исходный код

```
# Общий Containerfile
FROM docker.io/library/postgres:alpine3.22

ENV POSTGRES_USER=pozordom_user
ENV POSTGRES_PASSWORD=pozordom_pass
ENV POSTGRES_DB=pozordom

# Лаб 1 - Создание таблиц
COPY lab1/code/init.sql /docker-entrypoint-initdb.d/01_init.sql
# Лаб 2.1 - Наполнение таблиц и представления
COPY lab2_1/code/init_data.sql /docker-entrypoint-initdb.d/02_init_data.sql
COPY lab2_1/code/views.sql /docker-entrypoint-initdb.d/03_views.sql
# Лаб 3 - пользовательские функции и триггеры
COPY lab3/code/functions_triggers.sql /docker-entrypoint-initdb.d/04_functions_triggers.sql
```

Приложение A2. Функции и триггеры

```
-- functions_triggers.sql

-- 1. функция save_devices: вставка или обновление устройства
create or replace function save_devices(
    _id bigint,
    _hub_id int,
    _type_id int,
    _name varchar(100),
    _status jsonb
)
returns bigint as $$
declare
    used_id bigint;
begin
    if _id is null then
        insert into devices (hub_id, type_id, name, status)
        values (_hub_id, _type_id, _name, _status)
        returning id into used_id;
```

```

else
    update devices
    set hub_id = _hub_id,
        type_id = _type_id,
        name = _name,
        status = _status
    where id = _id;
    used_id := _id;
end if;
return used_id;
end;
$$ language plpgsql;

-----

-- 2. функция delete_hubs с проверкой внешних ссылок
alter table devices drop constraint if exists devices_hub_id_fkey;
alter table devices add constraint devices_hub_id_fkey
    foreign key (hub_id) references hubs(id) on delete restrict;

create or replace function delete_hubs(_id bigint)
returns void as $$
begin
    delete from hubs where id = _id;
exception
    when foreign_key_violation then
        raise exception 'невозможно выполнить удаление, так как есть внешние ссылки.';
end;
$$ language plpgsql;

-----

-- 3. функция фильтрации устройств по id
create or replace function filter_devices_by_id(min_id bigint)
returns setof devices as $$
begin
    return query

```

```
select * from devices
where id >= min_id
order by id;
end;
```

```
$$ language plpgsql;
```

```
-- 4. составной тип и функция фильтрации массива
```

```
drop type if exists device_info;
```

```
create type device_info as (
  id bigint,
  name varchar(100),
  type_name varchar(50),
  hub_name varchar(100)
);
```

```
create or replace function filter_device_array(
  arr device_info[],
  min_id bigint
)
returns device_info[] as $$
begin
  return array(
    select (id, name, type_name, hub_name)::device_info
    from unnest(arr)
    where id >= min_id
  );
end;
$$ language plpgsql;
```

```
-- 5. таблица логга и триггер
```

```
create table if not exists log_devices (
```

```

id bigserial primary key,
device_id bigint references devices(id),
change_time timestamp default now(),
old_name varchar(100),
new_name varchar(100)
);

create or replace function log_device_change()
returns trigger as $$
begin
    if tg_op = 'INSERT' then
        insert into log_devices (device_id, new_name)
        values (NEW.id, NEW.name);
    elsif tg_op = 'UPDATE' then
        insert into log_devices (device_id, old_name, new_name)
        values (NEW.id, OLD.name, NEW.name);
    end if;
    return NEW;
end;
$$ language plpgsql;

drop trigger if exists trigger_log_devices on devices;
create trigger trigger_log_devices
after insert or update on devices
for each row
execute function log_device_change();

```

```

-- 6. функция с динамическим SQL для получения значения столбца
create or replace function get_column_value(
    table_name text,
    column_name text,
    record_id bigint
)
returns text as $$
declare

```

```
    result text;
begin
    execute format(
        'select %I from %I where id = $1',
        column_name, table_name
    )
    into result
    using record_id;
    return result;
end;
$$ language plpgsql;
```