

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.

Основы DDL-запросов в PostgreSQL.

Отчёт по лабораторной работе №1

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 со схемами, пользователями и ролями в PostgreSQL;
- познакомиться с типами данных в PostgreSQL;
- освоить основные варианты DDL-запросов в PostgreSQL;
- закрепить знания по проектированию структуры реляционной БД;
- создать рабочий материал для следующих лабораторных работ.

Задание

1. Разработать структуру базы данных.
2. Создать нового пользователя и пустую БД. Подключиться к созданной БД.
3. Написать и выполнить SQL-скрипт, создающий таблицы с ограничениями и индексами согласно разработанной структуре БД.

ER-диаграмма

ER-диаграмма для разработанной структуры БД представлена на рисунке 1.

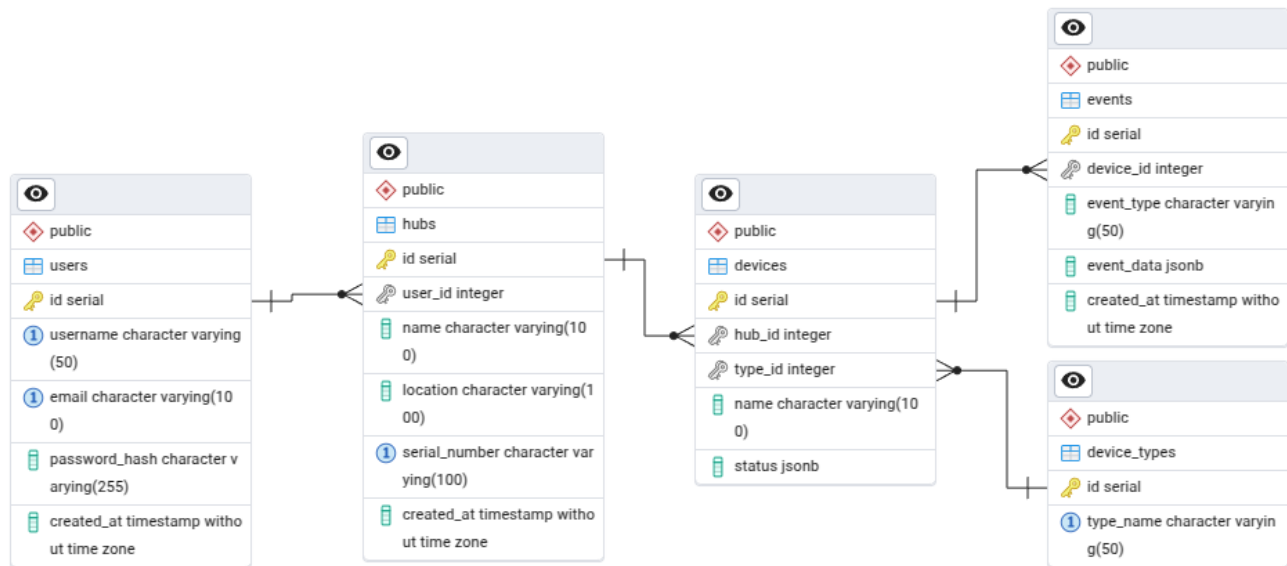


Рисунок 1 - ER-диаграмма.

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.

Краткое описание таблиц

1. Таблица **users**:

Содержит информацию о пользователях системы.

- **id** - уникальный идентификатор пользователя.
- **username** - уникальный логин пользователя.
- **email** - уникальная электронная почта.
- **password_hash** - хэш пароля.
- **created_at** - дата и время создания записи.

2. Таблица **hubs**:

Хранит информацию о хабах, к которым подключены устройства умного дома.

- **id** - уникальный идентификатор хаба.
- **user_id** - владелец хаба (связь с таблицей users).
- **name** - название хаба.
- **location** - местоположение хаба.
- **serial_number** - уникальный серийный номер хаба.
- **created_at** - дата и время создания записи.

3. Таблица **device_types**:

Содержит перечень типов устройств.

- **id** - уникальный идентификатор типа устройства.
- **type_name** - название типа устройства (например, "Лампа "Датчик").

4. Таблица **devices**:

Содержит информацию об устройствах, подключённых к хабам.

- **id** - уникальный идентификатор устройства.
- **hub_id** - связь с таблицей hubs.
- **type_id** - связь с таблицей device_types.
- **name** - название устройства.
- **status** - текущий статус устройства в формате JSON.

5. Таблица **events**:

Содержит события, генерируемые устройствами.

- **id** - уникальный идентификатор события.
- **device_id** - связь с таблицей devices.
- **event_type** - тип события (например, "включение "изменение температуры").
- **event_data** - данные события в формате JSON.
- **created_at** - дата и время создания записи.

Вывод

В ходе лабораторной работы были изучены пользователи, роли и типы данных в PostgreSQL. Была спроектирована структура базы данных для системы управления умным домом и создана соответствующая ER-диаграмма. Также выполнены SQL-скрипты для создания таблиц с ограничениями и индексами, включая таблицы пользователей, хабов, типов устройств, устройств и событий. В результате закреплены навыки проектирования и создания реляционной базы данных.

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. Исходный код

```
-- Таблица пользователей
create table users (
    id serial primary key,           -- Уникальный идентификатор пользователя
    username varchar(50) unique not null, -- Логин пользователя (уникальный)
    email varchar(100) unique not null,  -- Электронная почта (уникальная)
    password_hash varchar(255) not null, -- Хэш пароля
    created_at timestamp default now()   -- Дата и время создания записи
);

-- Таблица хабов
create table hubs (
    id serial primary key,           -- Уникальный идентификатор хаба
    -- Владелец хаба (связь с пользователем), при удалении пользователя хабы тоже удаляются
    user_id int not null references users(id) on delete cascade,
    name varchar(100) not null,      -- Название хаба
    location varchar(100),           -- Местоположение хаба
    serial_number varchar(100) unique not null, -- Серийный номер хаба (уникальный)
    created_at timestamp default now() -- Дата и время создания хаба
);
```

```
);
```

```
-- Таблица типов устройств (лампы, датчики и т.д.)
```

```
create table device_types (  
    id serial primary key,          -- Уникальный идентификатор типа устройства  
    type_name varchar(50) unique not null -- Название типа устройства  
);
```

```
-- Таблица устройств, подключённых к хабам
```

```
create table devices (  
    id serial primary key,          -- Уникальный идентификатор устройства  
    -- Связь с хабом, при удалении хаба устройство удаляется  
    hub_id int not null references hubs(id) on delete cascade,  
    -- Тип устройства, нельзя удалить тип если есть устройства этого типа  
    type_id int not null references device_types(id) on delete restrict,  
    name varchar(100) not null, -- Название устройства  
    status JSONB default '{} ' -- Текущий статус устройства в JSON {"power": "on"}  
);
```

```
-- Таблица событий, генерируемых устройствами
```

```
create table events (  
    id serial primary key,          -- Уникальный идентификатор события  
    -- Связь с устройством, при удалении устройства события удаляются  
    device_id int not null references devices(id) on delete cascade,  
    event_type varchar(50) not null, -- Тип события ("switch_on", "temperature_change")  
    event_data JSONB default '{}',   -- Данные события в JSON {"temperature": 22.5}  
    created_at timestamp default now() -- Дата и время создания события  
);
```