МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

По дисциплине: «Системы хранения и обработки данных»

Тема: «Развёртывание СУБД Postgres с использованием средств автоматизации развёртывания и управления приложениями»

Выполнил работу студент группы мИИВТ-251: Ивченко Я.А.

подпись, дата

Воронеж 2025

**Цель лабораторной работы:** изучить основы работы программного средства для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker на примере решения задачи развёртывания СУБД Postgres.

**Ход выполнения работы**

В первую очередь необходимо установить Docker Desktop на свой рабочий компьютор.

После было установлено средство для работы с СУБД DBeaver.

Создаем файл Dockerfike со следующим наполнением и попутно описываем что делает каждая из команд:

**# Базовый образ - используем официальный образ PostgreSQL последней версии**

**# Это основа, на которой будет строиться наш контейнер**

**FROM postgres:latest**

**# Устанавливаем переменную окружения для пароля базы данных**

**# Пароль будет 'dbpass' для пользователя PostgreSQL**

**ENV POSTGRES\_PASSWORD=dbpass**

**# Устанавливаем переменную окружения для имени пользователя базы данных**

**# Будет создан пользователь с именем 'dbuser'**

**ENV POSTGRES\_USER=dbuser**

**# Устанавливаем переменную окружения для имени базы данных**

**# Будет создана база данных с именем 'dbname'**

**ENV POSTGRES\_DB=dbname**

**# Копируем SQL-скрипт из папки init\_scripts на хосте (нашем компьютере)**

**# в специальную папку внутри контейнера /docker-entrypoint-initdb.d/**

**# Файлы в этой папке автоматически выполняются при первом запуске базы данных**

**COPY init\_scripts/init.sql /docker-entrypoint-initdb.d/init.sql**

Также необходимо создать SQL-скрипт, который автоматически выполнится при первом запуске базы.

**-- Создаем таблицу index\_mass в схеме public, если она еще не существует**

**-- Таблица будет содержать данные о пользователях: ID, вес и рост**

**CREATE TABLE IF NOT EXISTS public.index\_mass (**

**user\_id BIGINT, -- Уникальный идентификатор пользователя (большое целое число)**

**weight BIGINT, -- Вес пользователя в килограммах (большое целое число)**

**height BIGINT -- Рост пользователя в сантиметрах (большое целое число)**

**);**

**-- Вставляем тестовые данные в таблицу index\_mass**

**-- Добавляем трех пользователей с разными параметрами веса и роста**

**INSERT INTO public.index\_mass (user\_id, weight, height) VALUES**

**(1, 75, 175), -- Пользователь 1: вес 75 кг, рост 175 см**

**(2, 60, 182), -- Пользователь 2: вес 60 кг, рост 182 см**

**(3, 93, 181); -- Пользователь 3: вес 93 кг, рост 181 см**

Собираем образ как показано на рисунке 1: docker build -t d\_img:latest .

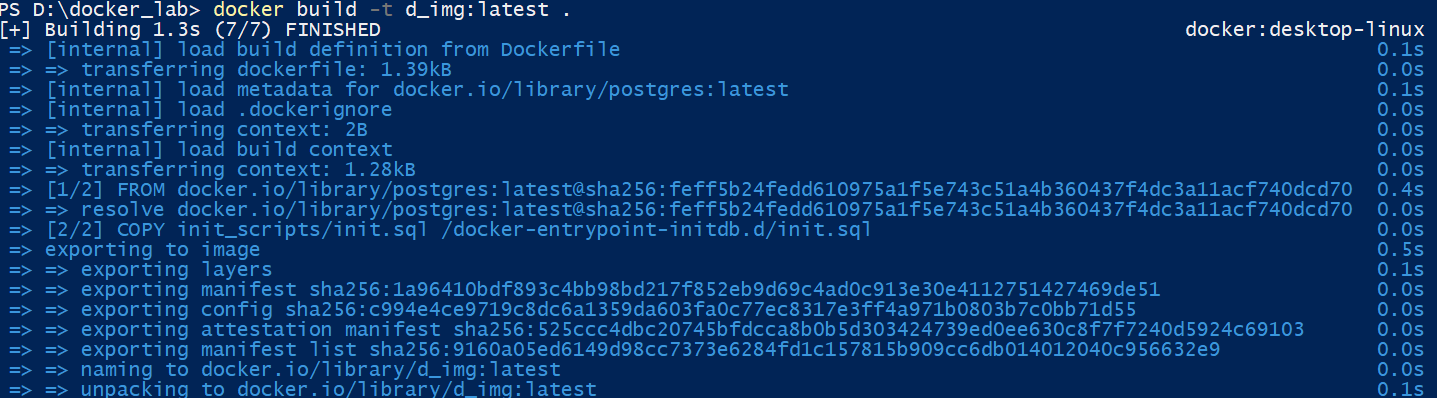


Рисунок 1 – Сборка образа

Далее необходимо подключиться к созданной БД через DBeaver. Для этого открываем графический интерфейс, создаем новое подключение и указываем параметры, которые задавались в SQL-скрипте. В результате на рисунке 2 можно увидеть подключенную БД.

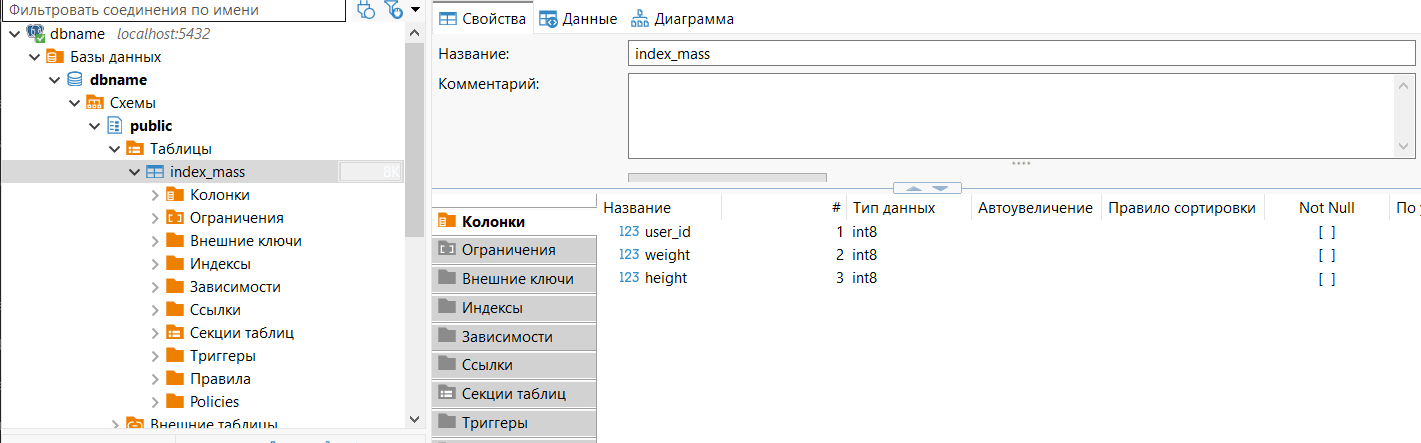


Рисунок 2 – Подключение БД в DBeaver

Создаем новый Docker-образ, но в этот раз с разделом для хранения информации с БД с помощью следующей команды

docker run -d -p 5432:5432 --name d\_cont\_vol -v d:\docker\_data:/var/lib/postgresql/data d\_img:latest

С помощью терминала выведем информацию о базе данных и убедимся, что все работает на рисунке 3.

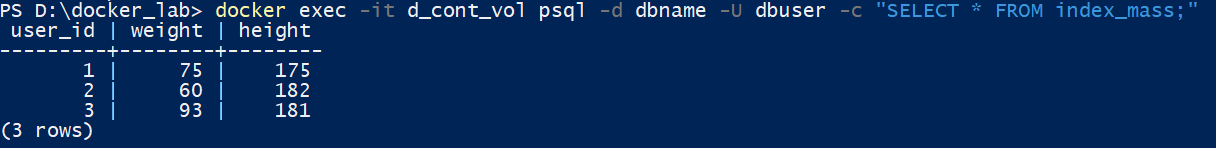


Рисунок 3 – Вывод информации о таблице

Для автоматизации и облегчения запоминания команд создадим файл docker-compose.

**services:**

**dbpost:**

**image: postgres:latest**

**environment:**

**POSTGRES\_DB: dbcompose # Новая база**

**POSTGRES\_USER: usrcompose # Новый пользователь**

**POSTGRES\_PASSWORD: passcompose # Новый пароль**

**volumes:**

**- ./init\_scripts/init.sql:/docker-entrypoint-initdb.d/init.sql**

**ports:**

**- "5433:5432" # Используем порт 5433 чтобы не конфликтовать**

Теперь вся инфраструктура запускается одной командой:

docker-compose up -d

Docker Compose сам создал вторую базу данных с новыми параметрами на порту 5433, чтобы избегать конфликта.

**Вывод**: во время выполнения лабораторной работы были изучены базовый команды и возможности Docker, была успешно развернута база postgressql с помощью Docker и docker-compose.

**Контрольные вопросы**

1. Docker — это платформа для разработки, доставки и запуска приложений в изолированных средах, называемых контейнерами.

2. Docker нужен, чтобы упаковать приложение со всеми его зависимостями в стандартный модуль. Это гарантирует, что приложение будет работать одинаково на любой системе, где есть Docker.

3. Docker-образ — это шаблон или «чертеж» с инструкциями для создания контейнера. Он включает в себя код, библиотеки и настройки.

4. Docker-контейнер — это запущенный экземпляр образа. Это изолированный процесс, работающий на основе своего образа.

5. Volume — это механизм для постоянного хранения данных, созданных и используемых контейнерами. Он нужен, чтобы сохранить информацию (например, файлы базы данных) после удаления контейнера.

6. Docker-compose — это инструмент для определения и управления многоконтейнерными приложениями. Все сервисы (например, веб-сервер и база данных) настраиваются в одном файле.

7. Разница между Dockerfile и docker-compose: Dockerfile содержит инструкции для сборки одного образа. Docker-compose.yml описывает, как запускать несколько контейнеров вместе как единое приложение.

8. Команда docker exec.

Она позволяет подключиться к уже запущенному контейнеру и выполнить внутри него любую команду.

9. Volume — это и есть инструмент для сохранения важных данных после остановки или удаления контейнера.