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 №8

По дисциплине: «Системы хранения и обработки данных»

Тема: «Разработка физической структуры базы данных»

Выполнил работу студент группы мИИВТ-241: Черкасов Н.Е.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Задачи лабораторной работы:

1) Изучить основы работы с СУБД PostgreSQL.

2) Изучить инструментарий для работы с СУБД (DBeaver или аналог).

3) Сформировать SQL-запросы для создания таблиц в СУБД Postgres, соответствующих логической структуре базы данных, разработанной в предыдущей лабораторной работе.

4) Сохранить SQL-запросы в соответствующем файле со скриптами SQL.

5) Описать каждый запрос (что делает запрос, из каких элементов (составляющих) состоит запрос, роль этих составляющих).

6) Обернуть созданную базу данных в docker-контейнер (файл docker-compose.yml).

7) Построить графическую диаграмму, отображающую физическую структуру базы данных. Сохранить диаграмму в графическом файле.

8) Описать диаграмму. Пояснить выбор сущностей, атрибутов, связей, ключей.

9) Подготовить отчёт о проделанной работе.

1. Для начала создания базы данных в левом меню кликнем правой кнопкой мыши по Databases и выберем Create и далее Database.

Зададим имя базе данных.

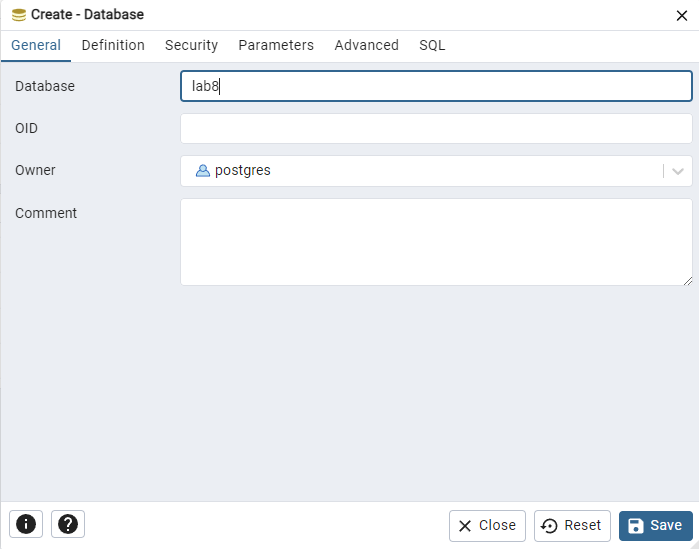


Рисунок 1 – Создание базы данных

Сохраним. Слева должна появиться наша база данных. Результат приведен на картинке 2.

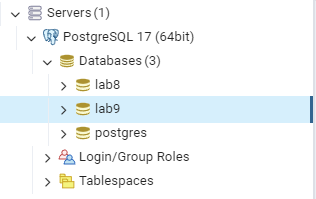


Рисунок 2 – Создание базы данных

Нажмем на созданную базу данных и выберем Querry Tool, тем самым перейдем в редактор кода для PgAdmin.

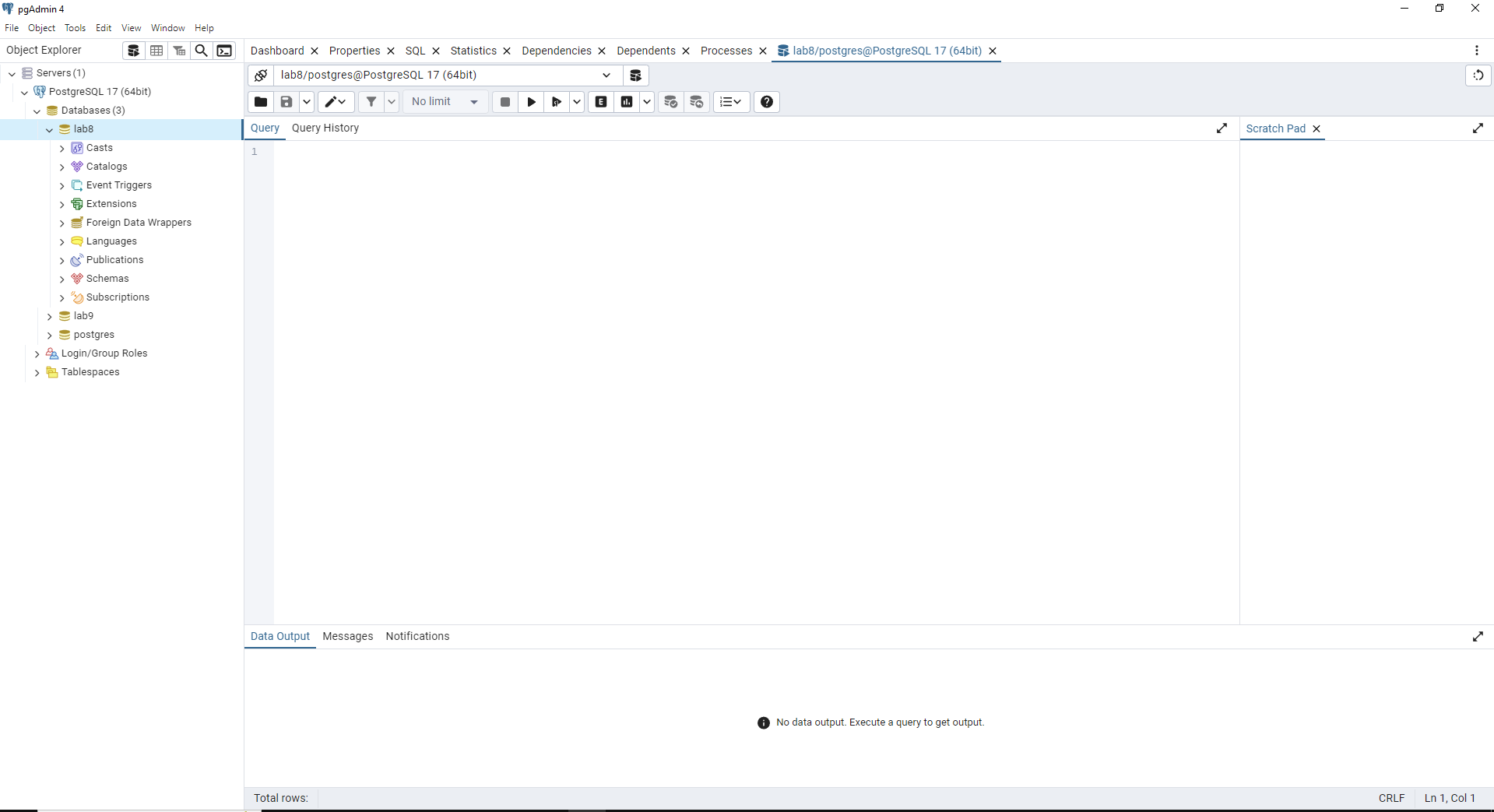


Рисунок 3 – Редактор Pgadmin

2. SQL скрипт.

CREATE TABLE IF NOT EXISTS Клиенты (

ID\_клиента SERIAL PRIMARY KEY,

ФИО\_клиента VARCHAR(255) NOT NULL,

Серия\_паспорта VARCHAR(10) NOT NULL,

Номер\_паспорта VARCHAR(10) NOT NULL,

Контактный\_номер VARCHAR(15) NOT NULL

);

CREATE TABLE IF NOT EXISTS Автомобили (

ID\_автомобиля SERIAL PRIMARY KEY,

Модель\_автомобиля VARCHAR(255) NOT NULL,

Цвет\_автомобиля VARCHAR(50) NOT NULL,

Год\_выпуска\_автомобиля INT NOT NULL,

Госномер\_автомобиля VARCHAR(20) NOT NULL,

Страховая\_стоимость\_автомобиля DECIMAL(10, 2) NOT NULL,

Стоимость\_одного\_дня\_проката DECIMAL(10, 2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Прокат (

ID\_проката SERIAL PRIMARY KEY,

ID\_клиента INT REFERENCES Клиенты(ID\_клиента) ON DELETE CASCADE,

ID\_автомобиля INT REFERENCES Автомобили(ID\_автомобиля) ON DELETE CASCADE,

Дата\_начала\_проката DATE NOT NULL,

Количество\_дней\_проката INT NOT NULL,

Общая\_стоимость DECIMAL(10, 2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Страхование (

ID\_страхования SERIAL PRIMARY KEY,

ID\_автомобиля INT REFERENCES Автомобили(ID\_автомобиля) ON DELETE CASCADE,

Номер\_полиса VARCHAR(50) NOT NULL,

Дата\_начала\_действия DATE NOT NULL,

Дата\_окончания\_действия DATE NOT NULL,

Страховая\_сумма DECIMAL(10, 2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Автомобили\_Клиенты (

ID SERIAL PRIMARY KEY,

ID\_клиента INT REFERENCES Клиенты(ID\_клиента) ON DELETE CASCADE,

ID\_автомобиля INT REFERENCES Автомобили(ID\_автомобиля) ON DELETE CASCADE

);

3. Описание запросов

1) CREATE TABLE - Создает таблицу для хранения информации.

2) IF NOT EXISTS – Проверяет существуют ли таблицы с данным названием, если таблица существует, SQL не выполняет команду создания.

3) Вводим названия строк.

4) Тип данных - Для идентификаторов используется SERIAL, что автоматически создает последовательность для генерации уникальных значений. Для текстовых полей используются VARCHAR, а для денежных значений — DECIMAL (10 – символов в числе, 2 – символов после запятой). DATE – тип данных для хранения даты.

5) Primary key – Первичный ключ, показывает уникальность записи.

6) References – Внешний ключ, устанавливает связь между таблицами.

7) ON DELETE CASCADE – При удалении записи из родительской таблицы, то удаляется запись в дочерней таблице.

4. Диаграмма.

Для визуализации наших запросов, необходимо нажать на иконку базы данных => затем выбрать ERD for Database. PgAdmin автоматически визуализирует ее.

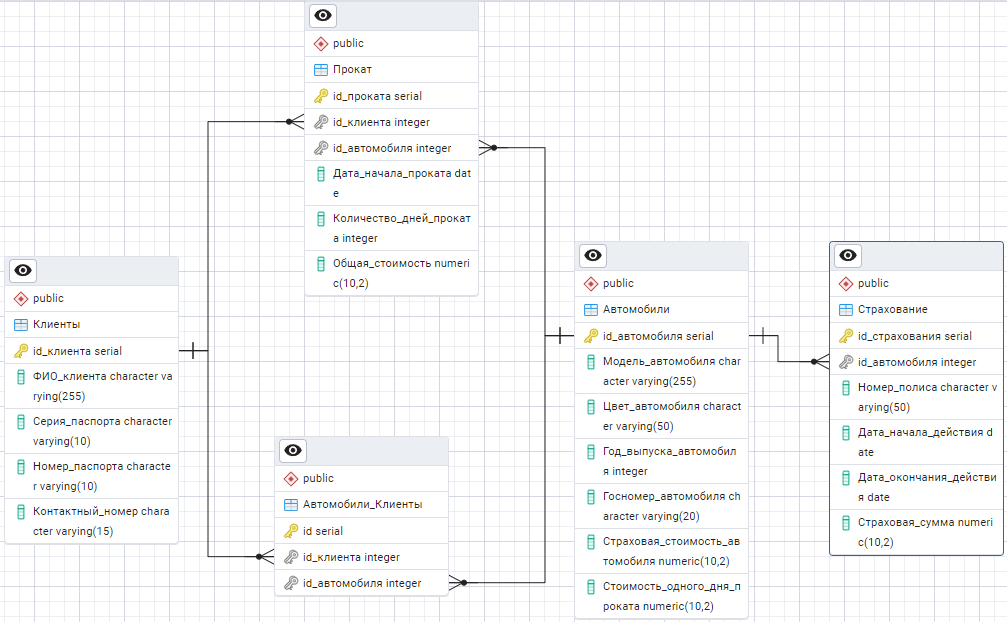


Рисунок 4 – Диаграмма базы данных

5. Связь таблиц.

1) Первичные ключи (PK):

- ID клиента (Клиенты)

- ID автомобиля (Автомобили)

- ID проката (Прокат)

- ID страхования (Страхование)

2) Внешние ключи (FK):

- ID клиента (Прокат) ссылается на ID клиента (Клиенты)

- ID автомобиля (Прокат) ссылается на ID автомобиля (Автомобили)

- ID автомобиля (Страхование) ссылается на ID автомобиля (Автомобили)

Первичные ключи уникально идентифицируют каждую запись в таблице, а внешние ключи обеспечивают связь между таблицами, что позволяет поддерживать целостность данных.

6. Типы связей.

* Связь «один ко многим» между клиентами и прокатом указывает на то, один клиент может иметь несколько записей о прокате, но каждая запись о прокате относится только к одному клиенту.
* Связь «один ко многим» между автомобилями и прокатом показывает, что один автомобиль может быть арендован разными клиентами в разное время.
* Связь «один ко многим» между автомобилями и страхованием позволяет иметь несколько полисов на один автомобиль.
* Связь «многие ко многим» между автомобилем и клиентом, так как если один клиент может арендовать несколько автомобилей, и один автомобиль может быть арендован несколькими клиентами.

7. YML формат таблиц.

Создаем файл docker-composse.yml. В файле указываем версию формата docker. Далее необходимо указать раздел Services, который будет показывать запущенные сервисы. Далее указываем имя сервиса; указываем, что будет использоваться Postgresql из образа Docker; environment: этот сервис задает переменные окружения для настройки базы данных:

POSTGRES\_DB: Имя базы данных;

POSTGRES\_USER: Имя пользователя для доступа к базе данных.

POSTGRES\_PASSWORD: Пароль для указанного пользователя.

volumes: Этот раздел определяет тома для хранения данных.

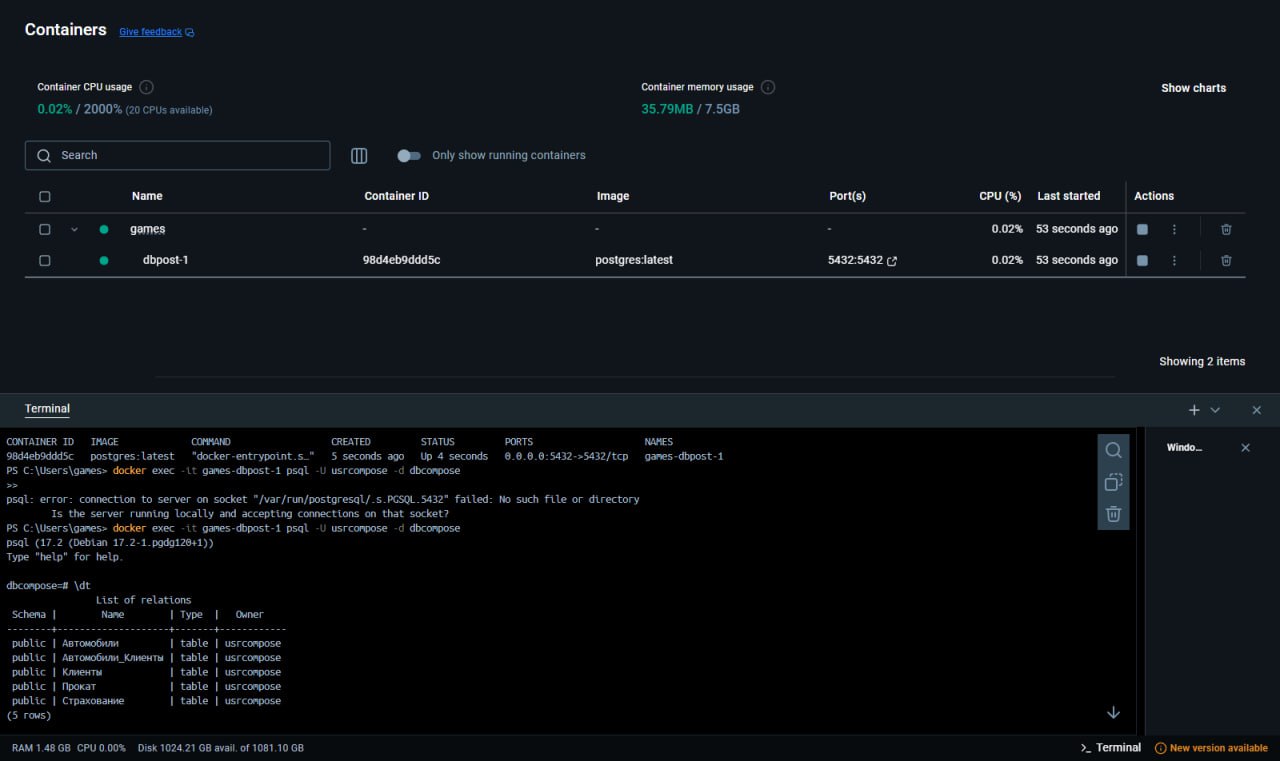


Рисунок 5 – Пример yml обертки базы данных

1) docker-compose up –d - команда запускает контейнеры, определенные в файле docker-compose.yml, в фоновом режиме (опция -d означает "detached mode"). Если контейнеры еще не созданы, они будут созданы автоматически. Если они уже существуют, они будут запущены заново.

2) docker exec -it games-dbpost-1 psql -U usrcompose -d dbcompose – эта команда выполняет команду внутри работающего контейнера Docker.

docker exec позволяет запускать команды в уже запущенном контейнере.

-it — это комбинация флагов, где -i означает "interactive", а -t — "pseudo-TTY", что позволяет вам взаимодействовать с командной строкой контейнера.

games-dbpost-1 — это имя контейнера, в котором вы хотите выполнить команду.

3) psql -U usrcompose -d dbcompose — это команда, которая запускает клиент PostgreSQL psql, используя пользователя usrcompose для подключения к базе данных dbcompose.

4) \dt. Данная команда, используемая в клиенте psql, которая отображает список всех таблиц в текущей базе данных. Показывает информацию о таблицах, таких как их имена и схемы.