

Data Engineering 0: основы баз данных и SQL - Семинар 2

Файлы docker-compose.yml и скрипты для БД доступны в репозитории курса в папке L02

Поднимаем локально Postgres

```
version: '3.8'
name: postgres

services:
  postgres:
    image: postgres:15-alpine
    container_name: postgres
    environment:
      - POSTGRES_USER=myuser          # Пользователь БД
      - POSTGRES_PASSWORD=mypassword # Пароль
      - POSTGRES_DB=mydatabase        # Имя базы данных
      - POSTGRES_HOST_AUTH_METHOD=trust # Разрешить локальные подключения без пароля (только для тестов!)
    ports:
      - "5432:5432"                  # Проброс портов: хост:контейнер
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U myuser -d mydatabase"]
      interval: 5s
      timeout: 5s
      retries: 5
    restart: unless-stopped

  init_user_postgres:
    image: postgres:15-alpine
    depends_on:
      - postgres:
          condition: service_healthy
    restart: "no"
    command: >
      bash -c "PGPASSWORD=postgres psql -h postgres -U myuser -d mydatabase -c 'ALTER ROLE myuser SUPERUSER'"

volumes:
  postgres_data:
```

Находясь в папке с файлом docker-compose.yml, открываем консоль.

```
[a.kaledin@macbook-C02CD2EAMD6P Преподавание – Базы данных % ls
DRIPDBLecture1.pdf      DRIPDBSeminar1.docx      docker-compose.yml
DRIPDBLecture1.pptx      DRIPDBSeminar1.pdf      kita_db_course
DRIPDBLecture2.pdf      DRIPDBSeminar2.docx      ~$IPDBSeminar1.docx
DRIPDBLecture2.pptx      DRIPJavaLecture1.pdf     ~$IPDBSeminar2.docx
DRIPDBSeminal2SqlQueries.sql  DRIPJavaLecture1.pptx  документы
a.kaledin@macbook-C02CD2EAMD6P Преподавание – Базы данных % ]
```

Из этой папки выполняем docker-compose up

```

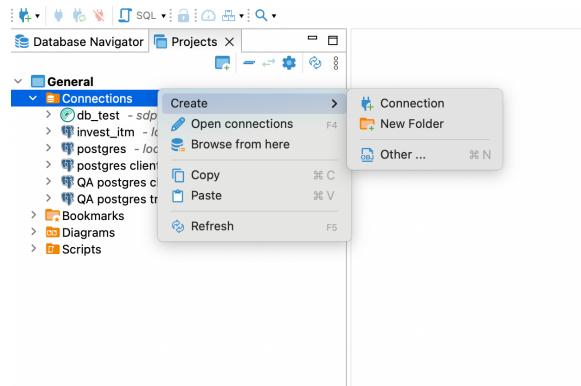
a.kaledin@macbook-C022CD2EAMD6P Преподавание - Базы данных % ls
DRIPDBlecture1.pdf          DRIPDBSeminar1.docx           docker-compose.yml
DRIPDBlecture1.pptx         DRIPDBSeminar1.pdf            kito_db_course
DRIPDBlecture1.pdf          DRIPDBSeminar2.docx           -$IPDBSeminar1.docx
DRIPDBlecture2.pptx         DRIPJavalecture1.pdf           -$IPDBSeminar2.docx
DRIPDBSeminal2SqlQueries.sql DRIPJavalecture1.pptx        документы
a.kaledin@macbook-C022CD2EAMD6P Преподавание - Базы данных % docker-compose up
WARN[0000] /Users/a.kaledin/hse/Преподавание - Базы данных/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
  ✓ Network postgres_default          Created             0.1s
  ✓ Container postgres               Created             0.1s
  ✓ Container postgres-init_user_postgres-1 Creat...      0.1s
Attaching to postgres, init_user_postgres-1
postgres
postgres      PostgreSQL Database directory appears to contain a database; Skipping initialization
postgres      2025-04-12 16:33:56.924 UTC [1] LOG:  starting PostgreSQL 15.12 on x86_64-pc-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit
postgres      2025-04-12 16:33:56.924 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
postgres      2025-04-12 16:33:56.924 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres      2025-04-12 16:33:56.938 UTC [29] LOG:  database system was shut down at 2025-04-12 10:54:52 UTC
postgres      2025-04-12 16:33:56.961 UTC [1] LOG:  database system is ready to accept connections
init_user_postgres-1 | ALTER ROLE
init_user_postgres-1 exited with code 0

```

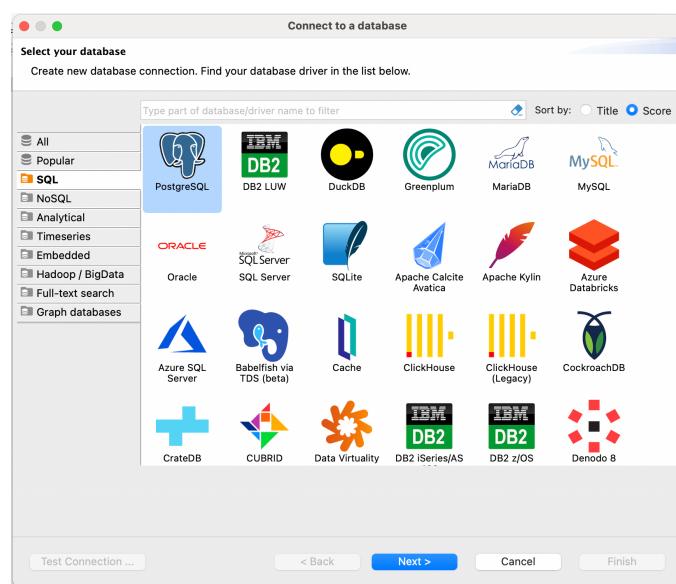
Ура, постгрес локально поднят! Можно подключиться к нему через консольные утилиты или с помощью программ вроде DBeaver или DataGrip.

Подключаемся к локальному Postgres через DBeaver

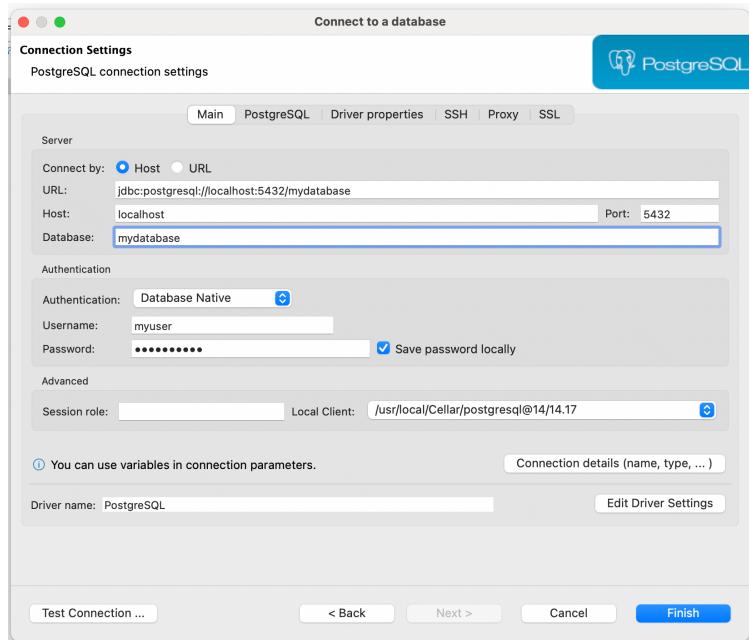
Открываем DBeaver и создаем новый Connection



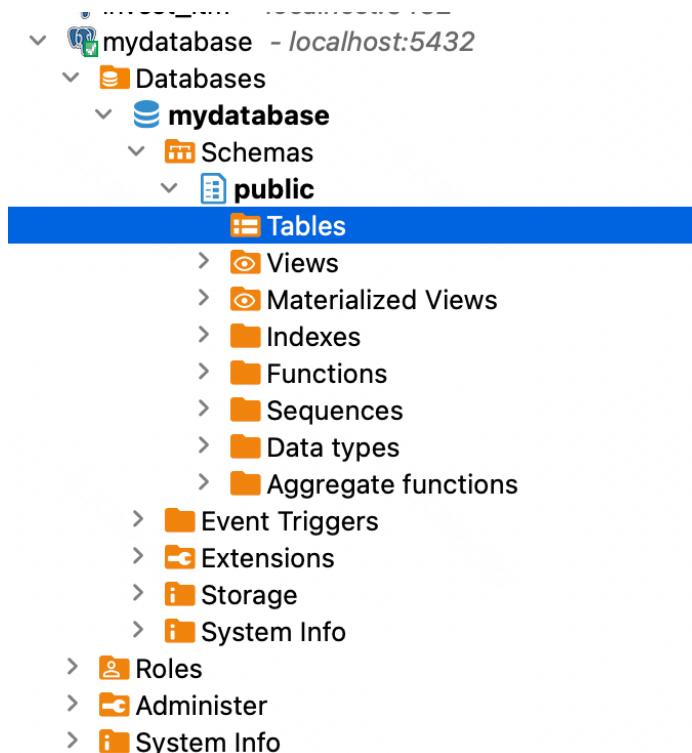
Выбираем драйвер для PostgreSQL



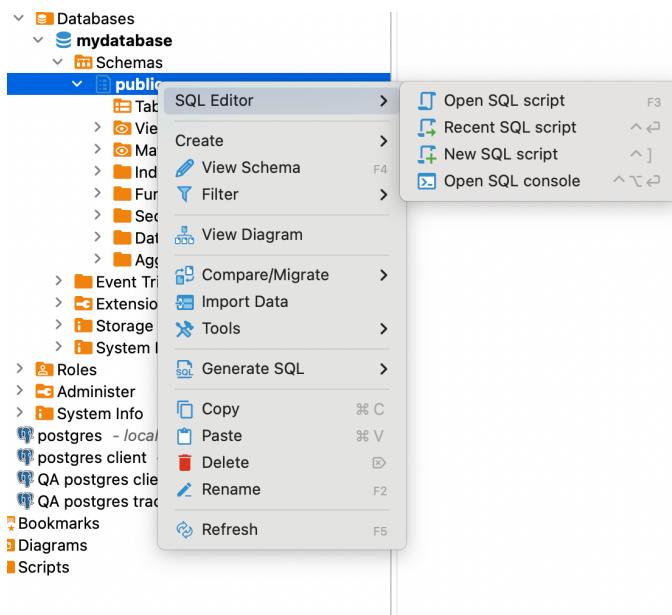
Задаем конфигурации для подключения по информации из docker-compose.yml файла



Ура, подключились к локальному PostgreSQL!



Открываем консоль для выполнения запросов



В консоли уже можем делать запросы. Но пока нет ни одной таблицы.

The screenshot shows the pgAdmin SQL console window titled '<mydatabase> Console'. The query 'SELECT 3*4;' is entered. The results grid shows a single row with one column labeled '?column?' containing the value '12'.

Создаем таблицу

The screenshot shows the pgAdmin SQL editor window titled 'mydatabase Console'. A complex SQL 'CREATE TABLE' statement is being typed, defining a 'users' table with columns for id, name, email, password, age, avatar, and created_at. The editor also displays the execution statistics at the bottom.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    age INTEGER,
    avatar VARCHAR(255),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Проверяем, что таблица была создана

The screenshot shows the pgAdmin interface. A query window titled '<mydatabase> Console' contains the SQL command: 'CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(255) UNIQUE NOT NULL, password VARCHAR(255) NOT NULL, age INTEGER, avatar VARCHAR(255), created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);'. Below it, a results window titled 'users 1' shows the table structure with columns: id, name, email, password, age, avatar, and created_at.

Добавляем какие-то записи в таблицу

The screenshot shows the pgAdmin interface. A query window contains the SQL command: 'INSERT INTO users (name, email, password, age, avatar) VALUES ('Иван Иванов', 'ivan@example.com', 'd131dd02c5e6eec4', 31, '/pictures/ivan.jpeg'), ('Мария Петрова', 'maria@example.com', '55ad340609f4b302', null, null), ('Алексей Смирнов', 'alex@example.com', 'd8823e3156348f5b', 15, '/root/alex.png'), ('Кек Лолов', 'kek@example.com', 'f131df02c5e6eec4', null, null), ('Эрик Картман', 'eric@southpark.com', '44ad440609f4b302', null, '/southpark/cartman.png');'. Below it, a statistics window titled 'Statistics 1' shows 'Updated Rows 5' with the same insert statement listed. The status bar at the bottom indicates 'Finish time Sat Apr 12 19:45:29 MSK 2025'.

Видим, что запрос изменил 5 строк. Проверяем, что теперь лежит в таблице

The screenshot shows the pgAdmin interface. A query window contains the SQL command: 'SELECT * FROM users;'. Below it, a results window titled 'users 1' displays the following data:

| | id | name | email | password | age | avatar | created_at |
|---|----|-----------------|--------------------|------------------|--------|------------------------|-------------------------|
| 1 | 1 | Иван Иванов | ivan@example.com | d131dd02c5e6eec4 | 31 | /pictures/ivan.jpeg | 2025-04-12 19:45:25.129 |
| 2 | 2 | Мария Петрова | maria@example.com | 55ad340609f4b302 | [NULL] | [NULL] | 2025-04-12 19:45:25.129 |
| 3 | 3 | Алексей Смирнов | alex@example.com | d8823e3156348f5b | 15 | /root/alex.png | 2025-04-12 19:45:25.129 |
| 4 | 4 | Кек Лолов | kek@example.com | f131df02c5e6eec4 | [NULL] | [NULL] | 2025-04-12 19:45:25.129 |
| 5 | 5 | Эрик Картман | eric@southpark.com | 44ad440609f4b302 | [NULL] | /southpark/cartman.png | 2025-04-12 19:45:25.129 |

Далее прикладываю весь список запросов, которые были использованы в рамках подготовки лекции 2. С ними можно играть, как вы хотите. При желании можете попробовать сломать что-нибудь в БД, задача интересная и безопасная, т.к. в случае проблем вам просто нужно будет потушить контейнер с БД и поднять его заново. Удачи в экспериментах!

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    age INTEGER,
    avatar VARCHAR(255),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
```

```
);

ALTER TABLE users
ADD COLUMN phone_number VARCHAR(15);

CREATE INDEX users_age_idx ON users (age);

INSERT INTO users (name, email, password, age, avatar)
VALUES
('Иван Иванов', 'ivan@example.com', 'd131dd02c5e6eec4', 31, '/pictures/ivan.jpeg'),
('Мария Петрова', 'maria@example.com', '55ad340609f4b302', null, null),
('Алексей Смирнов', 'alex@example.com', 'd8823e3156348f5b', 15, '/root/alex.png'),
('Кек Лолов', 'kek@example.com', 'f131df02c5e6eec4', null, null),
('Эрик Картман', 'eric@southpark.com', '44ad440609f4b302', null, '/southpark/cartman.png');

UPDATE users
SET PHONE_NUMBER = '88005553535'
WHERE email = 'eric@southpark.com';

DELETE FROM users
WHERE name = 'Алексей Смирнов';

SELECT 1+3;

SELECT id, name, age FROM users;

SELECT * FROM users
WHERE avatar IS NOT NULL;

SELECT * FROM users
WHERE avatar IS NOT NULL AND age > 20;

SELECT id, name, email, created_at FROM users
WHERE email LIKE '%@southpark.com';

SELECT id, name, created_at FROM users
ORDER BY id
LIMIT 2 OFFSET 0;

SELECT id, name, created_at FROM users
ORDER BY id
LIMIT 2 OFFSET 2;

ALTER TABLE users
ADD COLUMN company VARCHAR(15);

SELECT id, name, company FROM users;

SELECT company, count(*) FROM users
GROUP BY company;

SELECT id, name, avatar, age FROM users;

WITH users_with_avatar AS (
    SELECT id, name, age, email
    FROM users
    WHERE avatar IS NOT NULL
)
SELECT * FROM users_with_avatar
WHERE age > 20;

SELECT id, name, company FROM users;

CREATE TABLE companies (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

INSERT INTO companies (name)
VALUES
('hse'),
('meme factory'),
('southpark');
```

```
ALTER TABLE users
ADD COLUMN company_id INTEGER;

ALTER TABLE users
DROP COLUMN company;

SELECT id, name, created_at FROM users;

SELECT MAX(created_at) AS last_registered
FROM users;

SELECT id, name, age, company_id FROM users;

SELECT id, name, age, company_id, avg(age) OVER (PARTITION BY company_id) FROM users;

SELECT id, name, age, company_id FROM users;
SELECT id, name FROM COMPANIES ;

SELECT u.id, u.name AS user_name, c.name AS company_name
FROM users u
INNER JOIN companies c
ON u.company_id = c.id;

SELECT u.id, u.name AS user_name, c.name AS company_name
FROM users u
FULL JOIN companies c
ON u.company_id = c.id;

SELECT u.id, u.name AS user_name, c.name AS company_name
FROM users u
CROSS JOIN companies c;

SELECT u.id, u.name AS user_name, c.name AS company_name
FROM users u
FULL JOIN companies c
ON u.company_id = c.id
WHERE age > 10;

SELECT
c.name AS company_name,
COUNT(u.id) AS total_users,
ROUND(AVG(u.age)) AS avg_age,
(
    SELECT u2.name
    FROM users u2
    WHERE u2.company_id = c.id
    ORDER BY u2.created_at DESC
    LIMIT 1
) AS last_registered_user
FROM companies c
LEFT JOIN users u ON u.company_id = c.id
GROUP BY c.id
HAVING COUNT(u.id) > 0
ORDER BY total_users DESC;
```

Дз

В рамках подготовки ДЗ 1 вам нужно было спроектировать схему для интернет-магазина и сдать ее в формате PlantUML. Пришло время реализовать ее на практике!

В рамках ДЗ2 вам необходимо написать SQL запросы для создания описанной вами схемы. Это включает в себя создание таблиц, которые вы описали в рамках предыдущего ДЗ, и связей между ними (вороны лапки / Foreign Key). Дополнительно необходимо наполнить созданные таблицы данными (хватит 3-4 записей в каждой таблице).

Над этой схемой нужно сделать несколько select'ов с where фильтрацией, а также показать через запросы select + join наличие связи между вашими сущностями. К примеру, написать запрос, который показывает всех пользователей с его избранными товарами.

Напоминаю, что предполагается присутствование следующих сущностей:

- Товар
- Пользователь
- Категория товара
- Заказ
- Корзина
- Избранные товары у пользователя

Сдать ДЗ2 нужно в формате 2 файлов в ваш приватный гитхаб-репозиторий:

1. Текстовый файл со всеми вашими SQL-скриптами с расширением .txt / .sql
2. PDF файл со скринами запросов и результатов их выполнения + ваше описание, что это за запросы и на какие вопросы отвечают / зачем вы их выполнили