МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8  
  
По дисциплине: Системы хранения и обработки данных.

Тема: Разработка физической структуры базы данных  
 Выполнил работу студент группы мИИВТ-231 Киселев А.А. (подпись) Фамилия, инициалы

Принял \_\_\_\_\_\_\_\_\_\_\_ Короленко В.В  
 (подпись) Фамилия, инициалы

Защищена\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

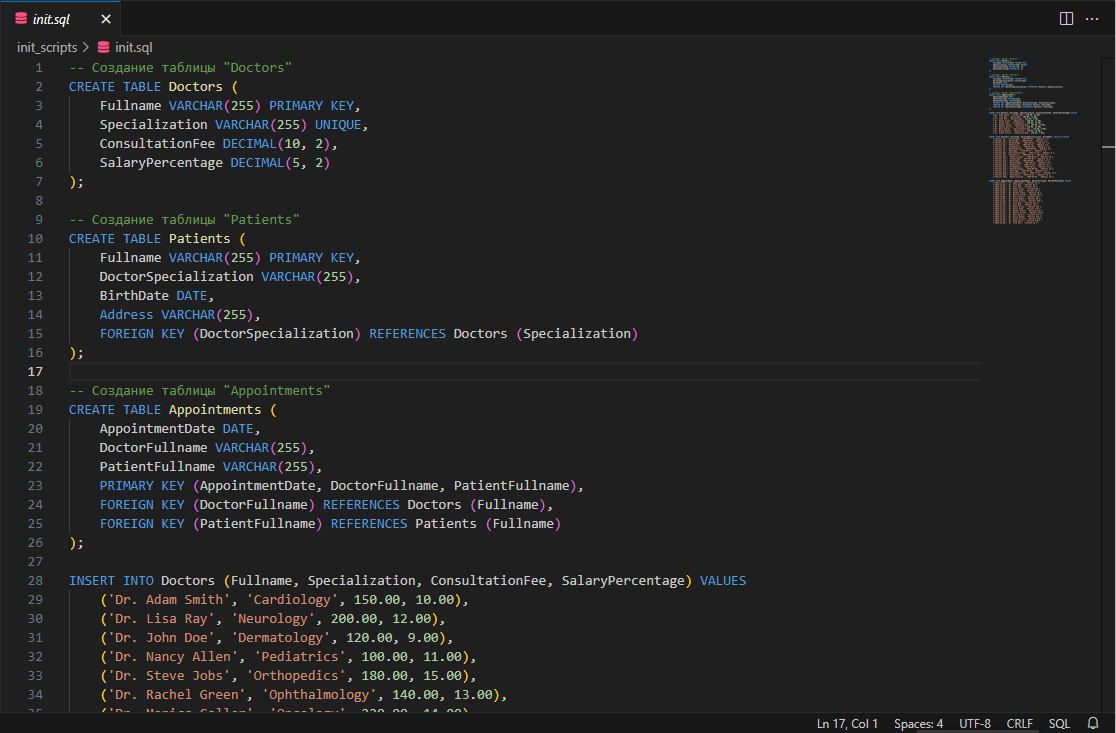
Воронеж 2023

**Цель лабораторной работы:** изучить и освоить процесс разработки физической структуры базы данных с использованием системы управления базами данных Postgres

**Основные задачи:**

* создание таблиц в СУБД Postgres с помощью SQL-запросов с атрибутами, связями, первичными и внешними ключами в соответствии с разработанной в предыдущей лабораторной работе логической структурой базы данных;
* построение графической диаграммы для отображения физической структуры базы данных.

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.**



1. **Сохранить SQL-запросы в соответствующем файле со скриптами SQL**
2. **Описать каждый запрос (что делает запрос, из каких элементов (составляющих) состоит запрос, роль этих составляющих).**

CREATE TABLE Doctors:

Создает таблицу "Doctors" с полями Fullname, Specialization, ConsultationFee и SalaryPercentage.

Fullname является первичным ключом, а Specialization уникальным.

ConsultationFee и SalaryPercentage представляют собой десятичные числа с фиксированной точностью.

CREATE TABLE Patients:

Создает таблицу "Patients" с полями Fullname, DoctorSpecialization, BirthDate и Address.

Fullname является первичным ключом.

DoctorSpecialization связано с полем Specialization в таблице "Doctors" через внешний ключ.

BirthDate представляет собой дату рождения пациента.

CREATE TABLE Appointments:

Создает таблицу "Appointments" с полями AppointmentDate, DoctorFullname и PatientFullname.

Составной первичный ключ состоит из AppointmentDate, DoctorFullname и PatientFullname.

DoctorFullname и PatientFullname связаны с соответствующими полями в таблицах "Doctors" и "Patients" через внешние ключи.

INSERT INTO Doctors:

Вставляет данные в таблицу "Doctors" с информацией о врачах, включая их полное имя, специализацию, консультационный гонорар и процент зарплаты.

INSERT INTO Patients:

Вставляет данные в таблицу "Patients" с информацией о пациентах, включая их полное имя, специализацию врача, дату рождения и адрес.

Специализация врача связана с полем Specialization в таблице "Doctors" через внешний ключ.

INSERT INTO Appointments:

Вставляет данные в таблицу "Appointments" с информацией о назначениях, включая дату, полное имя врача и полное имя пациента.

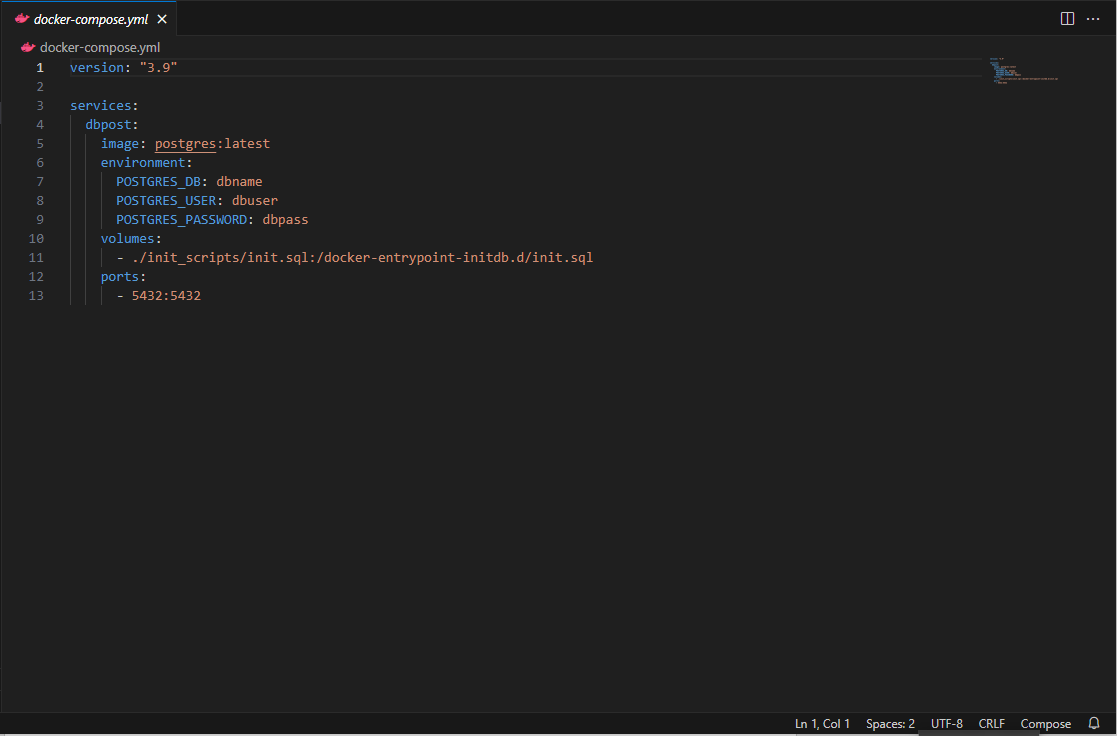
Использует внешние ключи для связи с таблицами "Doctors" и "Patients".

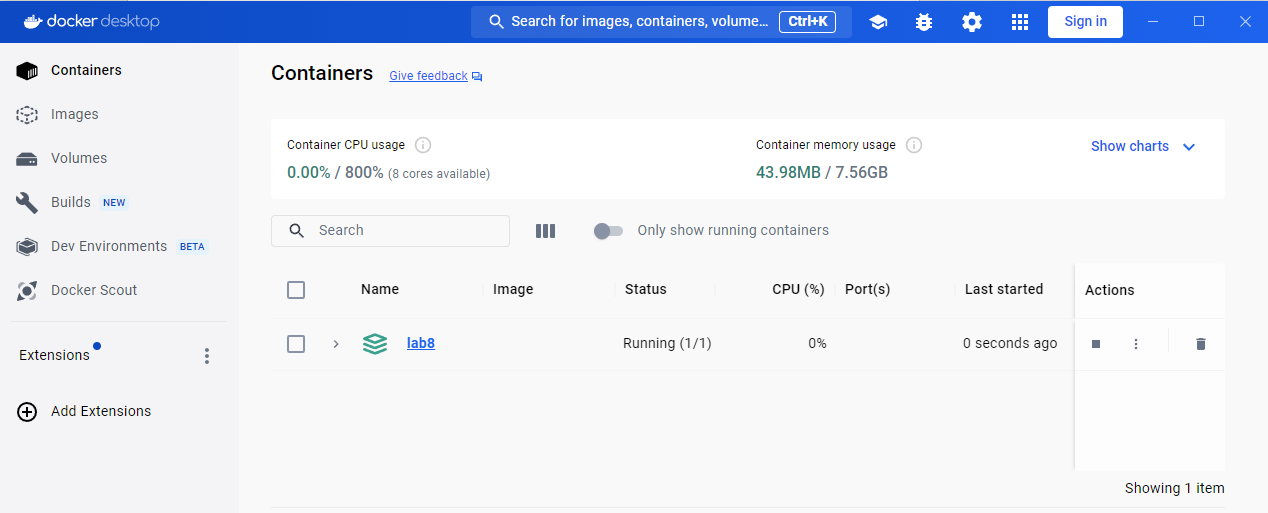
Ошибка (Failed to fetch):

Возвращает ошибку с сообщением "Failed to fetch". Это не является SQL-запросом, а скорее представляет собой ответ на неудачный запрос или отсутствие данных.

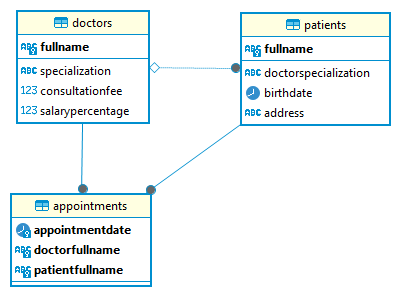
Все SQL-запросы выполняют действия по созданию таблиц, вставке данных и описанию связей между таблицами в базе данных для учета врачей, пациентов и назначений.

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





1. **Построить графическую диаграмму, отображающую физическую структуру базы данных. Сохранить диаграмму в графическом файле**



1. **Описать диаграмму. Пояснить выбор сущностей, атрибутов, связей, ключей**

Таблица "Doctors":

Сущность: Врачи.

Атрибуты: Fullname (Полное имя): VARCHAR(255) - используется в качестве первичного ключа (PRIMARY KEY), чтобы уникально идентифицировать каждого врача.

Specialization (Специализация): VARCHAR(255) - хранит информацию о специализации врача. Используется в качестве уникального атрибута (UNIQUE), чтобы каждая специализация была уникальной.

ConsultationFee (Стоимость консультации): DECIMAL(10, 2) - хранит информацию о стоимости консультации врача.

SalaryPercentage (Процент зарплаты): DECIMAL(5, 2) - хранит информацию о проценте зарплаты, который получает врач.

Таблица "Patients":

Сущность: Пациенты.

Атрибуты: Fullname (Полное имя): VARCHAR(255) - используется в качестве первичного ключа (PRIMARY KEY), чтобы уникально идентифицировать каждого пациента.

DoctorSpecialization (Специализация врача): VARCHAR(255) - хранит информацию о специализации врача, к которому относится пациент. Это внешний ключ (FOREIGN KEY), который ссылается на атрибут "Specialization" в таблице "Doctors".

BirthDate (Дата рождения): DATE - хранит информацию о дате рождения пациента.

Address (Адрес): VARCHAR(255) - хранит информацию об адресе пациента.

Таблица "Appointments":

Сущность: Записи на прием.

Атрибуты: AppointmentDate (Дата приема): DATE - хранит информацию о дате приема пациента.

DoctorFullname (Полное имя врача): VARCHAR(255) - хранит информацию о полном имени врача, к которому записан пациент. Это внешний ключ (FOREIGN KEY), который ссылается на атрибут "Fullname" в таблице "Doctors".

PatientFullname (Полное имя пациента): VARCHAR(255) - хранит информацию о полном имени пациента, который записан на прием. Это внешний ключ (FOREIGN KEY), который ссылается на атрибут "Fullname" в таблице "Patients".

Ключи:

PRIMARY KEY (AppointmentDate, DoctorFullname, PatientFullname) - используется для уникальной идентификации каждой записи на прием. Комбинация этих трех атрибутов обеспечивает уникальность записей.

Связи:

FOREIGN KEY (DoctorFullname) REFERENCES Doctors (Fullname) - устанавливает связь с таблицей "Doctors" по атрибуту "Fullname". Это означает, что значение атрибута "DoctorFullname" в таблице "Appointments" должно быть существующим значением в атрибуте "Fullname" таблицы "Doctors".

FOREIGN KEY (PatientFullname) REFERENCES Patients (Fullname) - устанавливает связь с таблицей "Patients" по атрибуту "Fullname". Это означает, что значение атрибута "PatientFullname" в таблице "Appointments" должно быть существующим значением в атрибуте "Fullname" таблицы "Patients".

Таким образом, в предложенной схеме базы данных учитываются сущности (врачи, пациенты, записи на прием), их атрибуты (полное имя, специализация, стоимость консультации и другие), связи (связь между таблицами "Doctors" и "Patients" через специализацию врача) и ключи (первичные и внешние ключи для уникальной идентификации записей и обеспечения связей между таблицами)

**Контрольные вопросы:**

1) PostgreSQL - это мощная объектно-реляционная система управления базами данных (СУБД), которая предоставляет расширенные возможности для хранения, организации и обработки данных. Вот некоторые особенности PostgreSQL:

- Поддержка SQL: PostgreSQL полностью совместим с языком SQL и предоставляет широкий набор функций и операторов для работы с данными.

- Расширяемость: PostgreSQL позволяет создавать пользовательские типы данных, функции, операторы и агрегатные функции, что позволяет адаптировать СУБД под конкретные потребности проекта.

- Поддержка транзакций: PostgreSQL обеспечивает ACID-свойства (атомарность, согласованность, изолированность, долговечность) для обработки транзакций, что гарантирует целостность данных.

- Расширенные возможности индексации: PostgreSQL предлагает различные типы индексов, включая B-деревья, хеш-индексы, GiST-индексы и другие, что повышает производительность запросов.

- Поддержка хранилищ JSON и геоданных: PostgreSQL имеет возможность хранить и обрабатывать данные в формате JSON и имеет встроенную поддержку геоданных с использованием расширения PostGIS.

2) DDL (Data Definition Language) - это язык определения данных, который используется для создания, изменения и удаления структуры базы данных. Он включает в себя команды для создания таблиц, индексов, представлений, процедур и других объектов базы данных. Некоторые примеры команд DDL в PostgreSQL включают CREATE, ALTER и DROP.

3) DML (Data Manipulation Language) - это язык манипулирования данными, который используется для вставки, обновления, удаления и выборки данных из таблиц базы данных. Он включает в себя команды для выполнения операций CRUD (Create, Read, Update, Delete) над данными. Примеры команд DML в PostgreSQL включают INSERT, UPDATE, DELETE и SELECT.

4) DCL (Data Control Language) - это язык управления данными, который используется для управления правами доступа и безопасностью базы данных. Он включает в себя команды для предоставления и отзыва прав доступа, управления пользователями и ролями, а также управления транзакциями. Примеры команд DCL в PostgreSQL включают GRANT, REVOKE и BEGIN/COMMIT/ROLLBACK.

5) Запросы в SQL-синтаксисе, как правило, начинаются с ключевого слова SELECT. Команда SELECT используется для выборки данных из таблицы или представления.

6) Команда CREATE DATABASE используется для создания базы данных в PostgreSQL. Пример использования команды CREATE DATABASE:

CREATE DATABASE имя\_базы\_данных;

7) Команда CREATE TABLE используется для создания таблицы в базе данных. Пример использования команды CREATE TABLE:

CREATE TABLE имя\_таблицы (

столбец1 тип\_данных,

столбец2 тип\_данных,

...

);

8) SELECT \* FROM employee WHERE sal IN (1500, 3000), SELECT \* FROM employee WHERE mod(mod(sal, 23), 5) = 0, SELECT \* FROM employee WHERE mod(sal, 23) = 0

9) Для достижения этой задачи, запрос SELECT \* FROM employee ORDER BY dept\_id NULLS FIRST; будет правильным выбором. Этот запрос упорядочит данные по столбцу "dept\_id" в порядке возрастания (ASC) и поместит строки без подразделения в начало списка (NULLS FIRST).

10) ‘a’

11) Неверными конструкциями запроса являются:

Select список полей from список таблиц union Select список полей from список таблиц order by поля для сортировки: UNION оператор не может использоваться в сочетании с ORDER BY второго SELECT выражения.

Select список полей from список таблиц having условия группировки group by поля для группировки: HAVING предложение используется для фильтрации группированных данных, но в данном случае оно появляется перед GROUP BY, что является неверным синтаксисом.

12) Для вычисления среднего объема продаж (advance) книг и годового дохода (sales) от продажи книг по всем категориям (type) из таблицы titles подходит запрос: Select avg(advance), sum(sales) from titles;. В данном случае нет необходимости в фильтрации по конкретным категориям, поэтому не указывается условие WHERE.