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 Ахлестин А.И. (подпись) Фамилия, инициалы

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

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

Воронеж 2023

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

Задание на лабораторную работу:

1. Создание таблиц в СУБД Postgres с помощью SQL-запросов;
2. Построение графической диаграммы физической структуры;

Ход выполнения задания:

# 1 Создание таблиц в СУБД Postgres с помощью SQL-запросов

# Формирование SQL-запросов для создания базы

# Сформируем запросы согласно заданию и структуре базы данных из предыдущей работы.

# Список получившихся запросов:

# -- Таблица Врачи

# CREATE TABLE Doctors (

# Doctor\_ID SERIAL PRIMARY KEY,

# Full\_Name VARCHAR(255) NOT NULL,

# Specialty VARCHAR(255),

# Percentage DECIMAL(5,2)

# );

# -- Таблица Пациенты

# CREATE TABLE Patients (

# Patient\_ID SERIAL PRIMARY KEY,

# Last\_Name VARCHAR(255) NOT NULL,

# First\_Name VARCHAR(255) NOT NULL,

# Middle\_Name VARCHAR(255),

# Birth\_Date DATE,

# Address VARCHAR(255)

# );

# -- Таблица Прием

# CREATE TABLE Appointments (

# Appointment\_ID SERIAL PRIMARY KEY,

# Doctor\_ID INTEGER REFERENCES Doctors(Doctor\_ID),

# Patient\_ID INTEGER REFERENCES Patients(Patient\_ID),

# Appointment\_Date DATE,

# Cost DECIMAL(8,2)

# );

# -- Таблица ПриемПациентов

# CREATE TABLE AppointmentsPatients (

# Appointment\_ID INTEGER REFERENCES Appointments(Appointment\_ID),

# Patient\_ID INTEGER REFERENCES Patients(Patient\_ID),

# PRIMARY KEY (Appointment\_ID, Patient\_ID)

# );

# -- Таблица Квитанция

# CREATE TABLE Receipts (

# Receipt\_ID SERIAL PRIMARY KEY,

# Appointment\_ID INTEGER REFERENCES Appointments(Appointment\_ID)

# );

# Сохранение запросов в файл со скриптами

# Теперь сохраним все получившиеся запросы в файл init.sql. Результат отображен на рисунке 1.

# 

# Рисунок 1 – Файл инициализации с запросами

# Описание всех запросов

# CREATE TABLE Doctors создает таблицу "Doctors" с четырьмя столбцами:

# Doctor\_ID - автоинкрементируемый идентификатор врача, является первичным ключом (PRIMARY KEY).

# Full\_Name - строка переменной длины (VARCHAR), представляющая полное имя врача. Не может быть пустым (NOT NULL).

# Specialty - строка переменной длины, представляющая специальность врача.

# Percentage - десятичное число (DECIMAL) с фиксированным числом знаков до и после запятой, представляющее процент (например, ставка).

# CREATE TABLE Patients создает таблицу "Patients" с шестью столбцами:

# Patient\_ID - автоинкрементируемый идентификатор пациента, является первичным ключом.

# Last\_Name, First\_Name, Middle\_Name - строки переменной длины, представляющие фамилию, имя и отчество пациента. Фамилия и имя не могут быть пустыми.

# Birth\_Date - дата рождения пациента.

# Address - строка переменной длины, представляющая адрес пациента.

# CREATE TABLE Appointments создает таблицу "Appointments" с пятью столбцами:

# Appointment\_ID - автоинкрементируемый идентификатор приема, является первичным ключом.

# Doctor\_ID - целочисленный столбец, который ссылается на Doctor\_ID в таблице "Doctors". Это внешний ключ, связывающий прием с врачом.

# Patient\_ID - целочисленный столбец, который ссылается на Patient\_ID в таблице "Patients". Это внешний ключ, связывающий прием с пациентом.

# Appointment\_Date - дата приема.

# Cost - десятичное число, представляющее стоимость приема.

# CREATE TABLE AppointmentsPatients создает таблицу "AppointmentsPatients" с двумя столбцами:

# Appointment\_ID - целочисленный столбец, который ссылается на Appointment\_ID в таблице "Appointments". Это внешний ключ, связывающий прием с приемом пациента.

# Patient\_ID - целочисленный столбец, который ссылается на Patient\_ID в таблице "Patients". Это внешний ключ, связывающий прием с пациентом.

# PRIMARY KEY (Appointment\_ID, Patient\_ID) - задает составной первичный ключ для таблицы, обеспечивающий уникальность комбинаций Appointment\_ID и Patient\_ID.

# CREATE TABLE Receipts создает таблицу "Receipts" с двумя столбцами:

# Receipt\_ID - автоинкрементируемый идентификатор квитанции, является первичным ключом.

# Appointment\_ID - целочисленный столбец, который ссылается на Appointment\_ID в таблице "Appointments". Это внешний ключ, связывающий квитанцию с приемом.

# Обертывание базы данных в контейнер

# Отредактируем файл docker-compose.yml, изменим имя базы данных и проверим правильность путей для скрипта инициализации. Отредактированный файл отображен на рисунке 2.

# 

# Рисунок 2 – Файл docker-compose.yml

# Далее откроем терминал и выполним команду для запуска контейнера в фоновом режиме: docker-compose up -d. Выполнение команды отображено на рисунке 3.

# 

# Рисунок 3 – Запуск контейнера

# Теперь проверим состояние контейнера(рисунок 4).

# 

# Рисунок 4 – Работающий контейнер

# Попробуем подключиться к базе данных, как показано на рисунке 5.

# 

# Рисунок 5 – Подключение к базе данных

# Проверим все таблицы(рисунок 6).

# 

# Рисунок 6 – Созданные таблицы

# 2 Построение графической диаграммы физической структуры

# Построим диаграмму физической структуры базы данных и опишем выбор сущностей, атрибутов, связей и ключей. Диаграмма представлена на рисунке 7.

# 

# Рисунок 7 – Диаграмма базы данных

# Выбор сущностей, атрибутов, связей и ключей в базе данных обусловлен логикой описываемой предметной области, в данном случае, системы управления медицинскими приемами.

# Таблица Врачи (Doctors):

# Атрибуты:

# Doctor\_ID: Уникальный идентификатор врача, необходим для однозначной идентификации.

# Full\_Name: Полное имя врача для удобства идентификации.

# Specialty: Информация о специализации врача.

# Percentage: Процентная ставка врача, что важно для расчетов стоимости приемов.

# Связи:

# Врач может провести множество приемов (один ко многим с Таблицей Прием).

# Ключи:

# Doctor\_ID: Первичный ключ, уникально идентифицирующий каждого врача.

# Таблица Пациенты (Patients):

# Атрибуты:

# Patient\_ID (PK): Уникальный идентификатор пациента.

# Last\_Name: Фамилия пациента.

# First\_Name: Имя пациента.

# Middle\_Name: Отчество пациента.

# Birth\_Date: Дата рождения пациента.

# Address: Адрес пациента.

# Связи:

# Пациент может пройти множество приемов (один ко многим с Таблицей Прием).

# Ключи:

# Patient\_ID: Первичный ключ.

# Таблица Прием (Appointments):

# Атрибуты:

# Appointment\_ID (PK): Уникальный идентификатор приема.

# Doctor\_ID (FK): Внешний ключ, связанный с Таблицей Врачи.

# Patient\_ID (FK): Внешний ключ, связанный с Таблицей Пациенты.

# Appointment\_Date: Дата проведения приема.

# Cost: Стоимость приема.

# Связи:

# Множество приемов может быть проведено одним врачом (многие к одному с Таблицей Врачи).

# Множество приемов может быть проведено одним пациентом (многие к одному с Таблицей Пациенты).

# Множество приемов может быть связано с множеством пациентов (многие ко многим с Таблицей ПриемПациентов).

# Ключи:

# Appointment\_ID: Первичный ключ.

# Таблица ПриемПациентов (AppointmentsPatients):

# Атрибуты:

# Appointment\_ID (FK): Внешний ключ, связанный с Таблицей Прием.

# Patient\_ID (FK): Внешний ключ, связанный с Таблицей Пациенты.

# Связи:

# Каждый прием связан с одним пациентом (один ко многим с Таблицей Пациенты).

# Каждый прием связан с одним приемом (один ко многим с Таблицей Прием).

# Ключи:

# Appointment\_ID, Patient\_ID: Составной ключ.

# Таблица Квитанция (Receipts):

# Атрибуты:

# Receipt\_ID (PK): Уникальный идентификатор квитанции.

# Appointment\_ID (FK): Внешний ключ, связанный с Таблицей Прием.

# Связи:

# Каждая квитанция связана с одним приемом (один ко многим с Таблицей Прием).

# Ключи:

# Receipt\_ID: Первичный ключ.

Вывод: в процессе выполнения работы было освоено формирование SQL-запросов и проектирование физической структуры базы данных.

Контрольные вопросы:

1. Особенности PostgreSQL:

PostgreSQL — объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом.

Поддерживает сложные SQL-запросы и множество расширений, включая хранимые процедуры и триггеры.

Поддерживает транзакции с высоким уровнем изоляции.

Может работать с различными типами данных, включая пользовательские типы.

Поддерживает многопоточность и параллелизм для обработки больших объемов данных.

Имеет расширенные возможности оптимизации запросов и индексирования.

Предоставляет множество дополнительных модулей и расширений, таких как PostGIS для геоданных.

1. Что такое DDL?

DDL — это язык определения данных.

Он используется для определения и изменения структуры базы данных, таких как создание, изменение и удаление таблиц, индексов и других объектов базы данных.

Примеры команд DDL включают CREATE, ALTER и DROP.

1. Что такое DML?

DML — это язык манипулирования данными.

Он используется для вставки, обновления и удаления данных в базе данных.

Примеры команд DML включают SELECT, INSERT, UPDATE и DELETE.

1. Что такое DCL?

DCL — это язык управления данными.

Он используется для управления правами доступа и безопасностью базы данных.

Примеры команд DCL включают GRANT (предоставление прав) и REVOKE (отзыв прав).

1. С какой команды, как правило, начинается запрос?

Запросы обычно начинаются с ключевого слова SELECT.

1. Какая команда используется для создания базы данных?

Команда CREATE DATABASE используется для создания базы данных.

1. Какая команда используется для создания таблицы в базе данных?

Команда CREATE TABLE используется для создания таблицы в базе данных.

1. Выбор сотрудников с empno 7844, 7788 и 7902:

SELECT \* FROM employee WHERE sal IN (1500, 3000).

1. Сортировка работников по подразделениям:

SELECT \* FROM employee ORDER BY dept\_id NULLS FIRST;

1. Какие значения подходят для типа char в PostgreSQL?

'1', 'a' и 'char'.

1. Какие конструкции запроса НЕ являются верными?

Select список полей from список таблиц union Select список полей from список таблиц order by поля для сортировки.

Select список полей from список таблиц order by поля для сортировки union Select список полей from список таблиц order by поля для сортировки.

1. Вычислить средний объем продаж (advance) книг и годовой доход (sales) от продажи книг по всем категориям (type) из таблицы titles:

Select avg(advance), sum(sales) from titles.