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: Тогушов В.А.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

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.

Учебная задача:

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.

10) Отчётные материалы загрузить в репозиторий Git и отправить ссылку на ваш репозиторий на платформе github на почту преподавателю. Репозиторий должен быть публичным.

Пакет отчётных документов в репозитории должен включать:

* отчёт;
* презентация;
* файл с SQL-скриптами для создания таблиц базы данных в СУБД Postgres;
* файл docker-compose.yml, с помощью которого запускается контейнер с созданной с помощью SQL-запросов базой данных в СУБД Postgres;

файл с диаграммой (.png или другой графический формат).

Индивидуальное задание:

Выполнить все пункты раздела «Учебная задача», оформить отчёт, представить результаты выполнения лабораторной работы к защите.

Сформируем SQL-запросы для создания таблиц в СУБД Postgres, соответствующих логической структуре базы данных, разработанной в предыдущей лабораторной работе:

Таблица «Квартиры»:

CREATE TABLE Квартиры (

ID\_квартиры SERIAL PRIMARY KEY,

Улица VARCHAR(255) NOT NULL,

Номер\_дома INTEGER NOT NULL,

Номер\_квартиры INTEGER NOT NULL,

Площадь FLOAT NOT NULL,

Количество\_комнат INTEGER NOT NULL,

UNIQUE (улица, номер\_дома, номер\_квартиры) -- Ограничение уникальности

);

Таблица «Риэлторы»:

CREATE TABLE Риэлторы (

ID\_риэлтора SERIAL PRIMARY KEY,

ФИО VARCHAR(255) NOT NULL,

Процент\_вознаграждения FLOAT NOT NULL

);

Таблица «Клиенты»:

CREATE TABLE Клиенты (

ID\_клиента SERIAL PRIMARY KEY,

ФИО VARCHAR(255) NOT NULL,

Контактная\_информация VARCHAR(255) NOT NULL

);

Таблица «Сделки»:

CREATE TABLE Сделки (

ID\_сделки SERIAL PRIMARY KEY,

Дата\_сделки DATE NOT NULL,

Цена\_квартиры FLOAT NOT NULL,

ID\_квартиры INTEGER NOT NULL,

ID\_риэлтора INTEGER NOT NULL,

FOREIGN KEY (ID\_квартиры) REFERENCES Квартиры (ID\_квартиры),

FOREIGN KEY (ID\_риэлтора) REFERENCES Риэлторы (ID\_риэлтора)

);

Таблица «Сделки-Клиенты»:

CREATE TABLE Сделки\_Клиенты (

ID SERIAL PRIMARY KEY,

ID\_сделки INTEGER NOT NULL,

ID\_клиента INTEGER NOT NULL,

Роль\_клиента VARCHAR(255) NOT NULL,

FOREIGN KEY (ID\_сделки) REFERENCES Сделки (ID\_сделки),

FOREIGN KEY (ID\_клиента) REFERENCES Клиенты (ID\_клиента),

UNIQUE (ID\_сделки, ID\_клиента)

);

Опишем запросы:

CREATE TABLE – создание таблицы.

SERIAL PRIMARY KEY – создание автоинкрементного поля (первичного ключа), уникального для каждой записи.

FOREIGN KEY ... REFERENCES – внешние ключи, обеспечивающие связь с другой таблицей через ключевое слово REFERENCES.

UNIQUE – ограничение уникальности на несколько столбцов (например, улица, номер дома, номер квартиры).

NOT NULL – указание, что значение обязательно для заполнения.

Сохраним SQL-запросы в соответствующем файле со скриптами SQL (Рисунок 1).

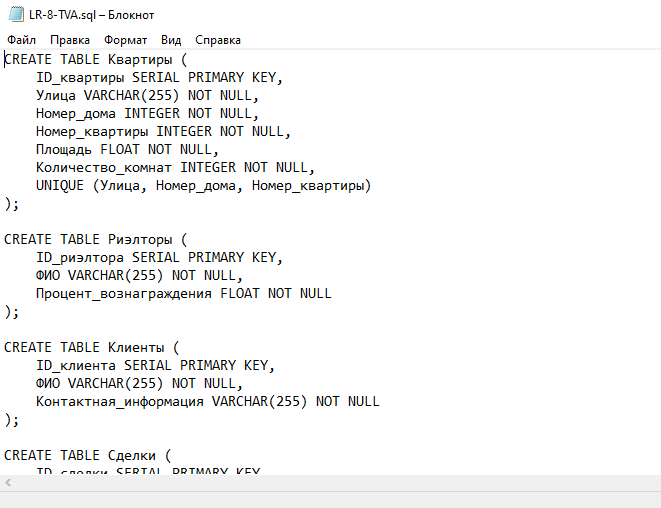


Рисунок 1 – Файл sql с запросами для создания таблиц базы данных

Теперь обернем созданную базу данных в docker-контейнер с помощью файла «docker-compose.yml» (Рисунок 2).

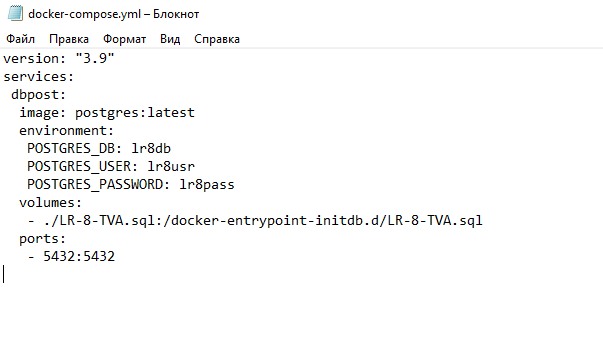


Рисунок 2 – Файл «docker-compose.yml»

В терминале запустим контейнер с помощью команды: docker-compose up.

Перейдем в DBeaver и подключимся к созданной базе данных, с помощью указанных в файле «docker-compose.yml» названия этой базы, имени пользователя и пароля (Рисунок 3).

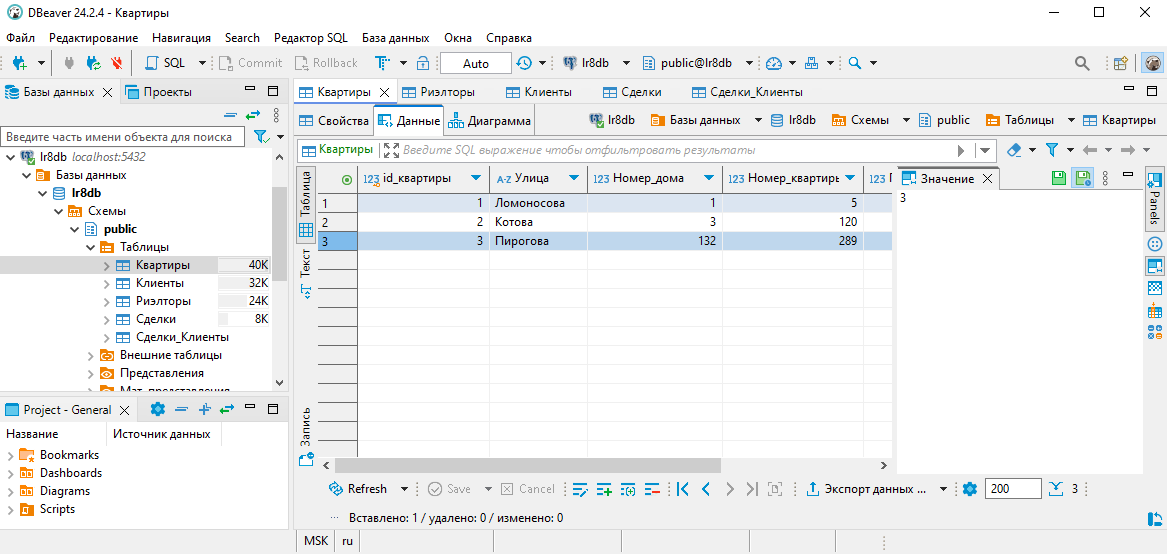


Рисунок 3 – Подключение к базе данных

Далее построим графическую диаграмму, отображающую физическую структуру базы данных (Рисунок 4).

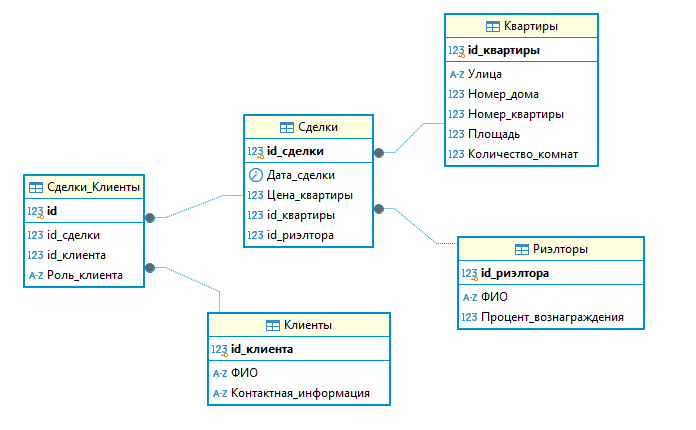


Рисунок 4 – Графическая диаграмма физической структуры базы данных

Описание диаграммы:

Таблица "Квартиры": хранит информацию о квартирах, которые участвуют в сделках.

Атрибуты:

id\_квартиры (PK) – уникальный идентификатор квартиры.

Улица – название улицы, где находится квартира.

Номер\_дома – номер дома.

Номер\_квартиры – номер квартиры.

Площадь – площадь квартиры в квадратных метрах.

Количество\_комнат – количество комнат в квартире.

Таблица позволяет однозначно идентифицировать каждую квартиру с помощью уникального идентификатора, а комбинация полей Улица, Номер\_дома и Номер\_квартиры делает данные уникальными.

Таблица "Риэлторы": хранит информацию о риэлторах, которые работают с недвижимостью.

Атрибуты:

id\_риэлтора (PK) – уникальный идентификатор риэлтора.

ФИО – полное имя риэлтора.

Процент\_вознаграждения – фиксированный процент, используемый для расчета комиссионных.

Уникальный идентификатор id\_риэлтора обеспечивает однозначность, а процент вознаграждения важен для расчетов.

Таблица "Клиенты": содержит информацию о клиентах, участвующих в сделках.

Атрибуты:

id\_клиента (PK) – уникальный идентификатор клиента.

ФИО – полное имя клиента.

Контактная\_информация – способ связи с клиентом (телефон, email и т.д.).

Каждого клиента можно идентифицировать по уникальному идентификатору, а контактная информация важна для взаимодействия.

Таблица "Сделки": учет информации о сделках купли-продажи недвижимости.

Атрибуты:

id\_сделки (PK) – уникальный идентификатор сделки.

Дата\_сделки – дата оформления сделки.

Цена\_квартиры – стоимость квартиры на момент сделки.

id\_квартиры (FK) – ссылка на таблицу «Квартиры».

id\_риэлтора (FK) – ссылка на таблицу «Риэлторы».

Связи:

Связана с таблицей «Квартиры» через id\_квартиры (Одна квартира участвует в нескольких сделках).

Связана с таблицей «Риэлторы» через id\_риэлтора (Один риэлтор может проводить множество сделок).

Таблица обеспечивает учёт всех сделок с указанием их участников и деталей.

Таблица "Сделки\_Клиенты": учет связей между клиентами и сделками.

Атрибуты:

id (PK) – уникальный идентификатор записи.

id\_сделки (FK) – ссылка на таблицу «Сделки».

id\_клиента (FK) – ссылка на таблицу «Клиенты».

Роль\_клиента – роль клиента в сделке (покупатель, продавец и т.д.).

Связи:

Связывает таблицу «Сделки» с таблицей «Клиенты» через уникальный идентификатор.

Поддерживает связь "многие ко многим".

Позволяет учитывать случаи, когда в одной сделке участвуют несколько клиентов с разными ролями.

Контрольные вопросы:

1) Особенности PostgreSQL.

Некоторые основные особенности PostgreSQL:

- Расширяемость. Пользователь может настроить систему, определив новые функции, типы, языки, агрегаты, индексы и операторы.

- Высокая производительность. В PostgreSQL используются индексы, гибкая система блокировок и интеллектуальный планировщик запросов.

- Работа с различными типами данных. В системе поддерживаются численные, булевые, символьные, составные, сетевые типы данных. Кроме того, PostgreSQL работает с перечислением, типами «дата/время», геометрическими примитивами, массивами, XML- и JSON-данными.

- Надежность. Система соответствует принципам ACID (атомарность, изолированность, непротиворечивость, сохранность данных). PostgreSQL быстро восстанавливается после сбоев благодаря журналу изменений транзакций (WAL).

- Поддержка стандарта SQL. Это значит, что команды и запросы для других баз данных будут работать и в PostgreSQL.

- Поддержка JSON и XML. В PostgreSQL можно хранить данные и управлять ими в форматах JSON и XML, поэтому её часто используют для работы с неструктурированными данными.

2) Что такое DDL?

DDL (Data Definition Language) – это группа команд, которые используются для создания и изменения структуры объектов базы данных: таблиц, представлений, схем и индексов.

Эти запросы позволяют настраивать базу данных – создавать с нуля и прописывать её структуру.

Некоторые известные команды SQL DDL: CREATE, ALTER, DROP.

3) Что такое DML?

DML (Data Manipulation Language) – это группа операторов, которые позволяют получать и изменять записи, присутствующие в таблице.

Эти запросы помогают добавлять, обновлять, удалять и выбирать данные. Примеры DML-запросов: SELECT, UPDATE, DELETE, INSERT.

4) Что такое DCL?

DCL (Data Control Language) – это подмножество языка управления базами данных SQL, предназначенное для осуществления административных операций, присваивающих или отменяющих право (привилегию) использовать базу данных, таблицы и другие объекты базы данных, а также выполнять те или иные операторы SQL.

Наиболее известные команды DCL: GRANT (для предоставления прав доступа к базе данных) и REVOKE (для отзыва ранее выданных прав).

5) С какой команды, как правило, начинается запрос?

Все инструкции SQL начинаются с ключевого слова. Это может быть SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW и другие.

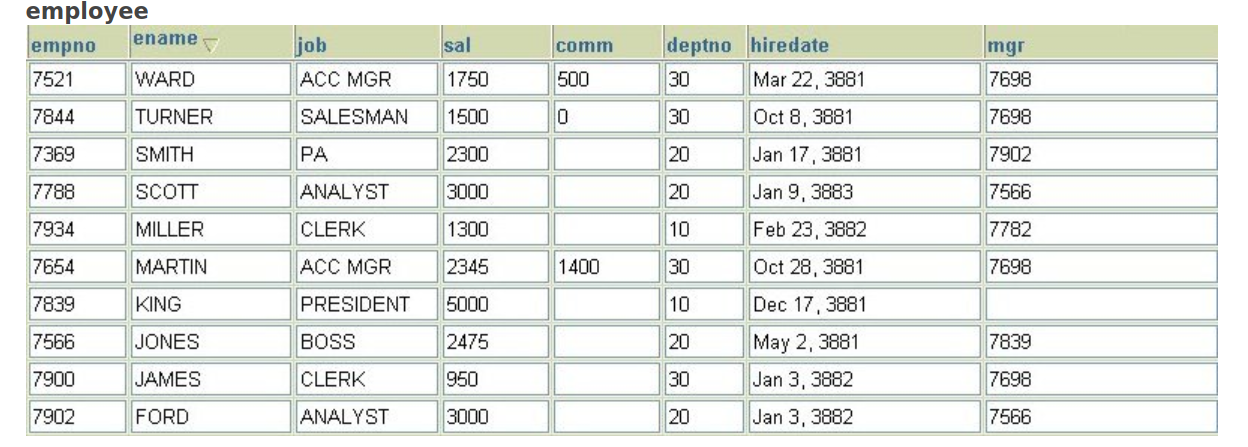
6) Какая команда используется для создания базы данных?

Для создания базы данных используется команда CREATE DATABASE.

7) Какая команда используется для создания таблицы в базе данных?

Для создания новой таблицы в базе данных используется команда CREATE TABLE.

8)



Из таблицы employee необходимо показать сотрудников с empno 7844, 7788 и 7902 одним запросом. Какие из приведенных ниже запросов соответствуют данному условию?

SELECT \* FROM employee WHERE mod(mod(sal, 23), 5) = 0;

SELECT \* FROM employee WHERE mod(sal, 23) = 0;

SELECT \* FROM employee WHERE sal = 3000;

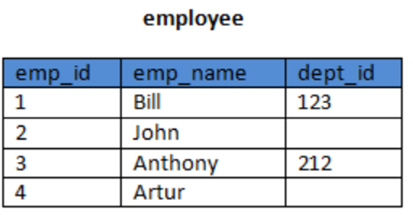
SELECT \* FROM employee WHERE sal = 1500;

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

Соответствует условию запрос:

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

9) В базе данных компании хранятся данные о работниках и подразделениях, к которым они относятся.



Ваша задача: достать весь список работников, при этом все работники без подразделения должны быть в начале списка. Какой из представленных запросов выполнит эту задачу?

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

SELECT \* FROM employee ORDER BY dept\_id ASC;

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

SELECT \* FROM employee ORDER BY dept\_id;

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

Задачу выполнит запрос:

SELECT \* FROM employee ORDER BY dept\_id NULLS FIRST; наиболее подходит, так как явно указывает расположение NULL значений в начале списка.

10) Какие из следующих значений подходят для типа char в PostgreSQL?

12, '1', 1.22, 'a' или 'char'

Для типа char подходят значения, приведенные к строке и не более одно символа, поэтому подходят значения '1' и 'a'.

'char' подойдет для типа char(n), где явно указывается длина строки.

11) Какие из перечисленных конструкций запроса НЕ являются верными? Выбрать можно несколько вариантов ответа.

- Select список полей from список таблиц where условия order by поля для сортировки

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

- Select список полей from список таблиц having условия группировки group by поля для группировки

- Select список полей from список таблиц where условия group by поля для группировки 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

Неверные запросы:

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

ORDER BY должен применяться ко всему результату объединения (UNION), а не только ко второй части.

- Select список полей from список таблиц having условия группировки group by поля для группировки

Ключевое слово HAVING используется после GROUP BY для фильтрации результатов группировки.

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

Как и в случае с вариантом 2, нельзя использовать несколько ORDER BY внутри объединения (UNION). ORDER BY может быть только один и должен находиться в конце всего объединенного результата.

12) Необходимо вычислить средний объем продаж (advance) книг и годовой доход (sales) от продажи книг по всем категориям (type) из таблицы titles. Какой из перечисленных фрагментов кода позволит решить поставленную задачу?

- Select avg (advance), sum (sales) from titles where type between ’business’ or ’internet’

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

- Select avg (advance), sum (sales) from titles where type between ’business’ and ’internet’

- Select avg (advance), sum (sales) from titles group by type

- Select avg (advance), sum (sales) from titles where type = ’business’ and type = ’internet’

Подходит запрос:

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