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: Власов И.В.

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.

Ход выполнения лабораторной работы

Для создания базы данных переходим в консоль в каталоге контейнера docker-compose, в нем выполняем команду подключения к базе данных.

docker exec -it shiod\_lab6-dbpost-1 psql -d dbcompose -U usrcompose. На рисунке 1 представлен результат ее работы:

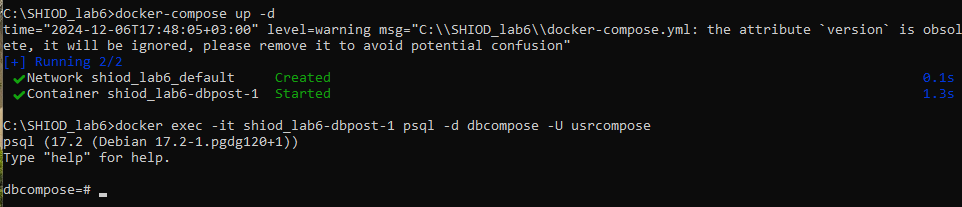


Рисунок 1 – Подключение к БД

Изменим файл docker-compose.yml, изменив в нем название .sql файла для создания таблиц на LR\_8.sql. В этом файле мы впишем скрипт для создания всех таблиц. Его содержание представлено ниже:

-- Создание таблицы "Издания"

CREATE TABLE Издания (

индекс\_издания SERIAL PRIMARY KEY,

название\_издания VARCHAR(255) NOT NULL,

вид\_издания VARCHAR(50) NOT NULL,

цена\_1\_экземпляра DECIMAL(10, 2) NOT NULL

);

-- Создание таблицы "Получатели"

CREATE TABLE Получатели (

идентификатор\_получателя SERIAL PRIMARY KEY,

фио\_подписчика VARCHAR(255) NOT NULL,

улица VARCHAR(255) NOT NULL,

номер\_дома VARCHAR(10) NOT NULL,

номер\_квартиры VARCHAR(10) NOT NULL

);

-- Создание таблицы "Подписки"

CREATE TABLE Подписки (

идентификатор\_подписки SERIAL PRIMARY KEY,

идентификатор\_получателя INT,

индекс\_издания INT,

дата\_начала\_подписки DATE NOT NULL,

срок\_подписки INT NOT NULL, -- Количество месяцев

стоимость\_подписки DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (идентификатор\_получателя) REFERENCES Получатели(идентификатор\_получателя),

FOREIGN KEY (индекс\_издания) REFERENCES Издания(индекс\_издания)

);

-- Создание таблицы "Доставка"

CREATE TABLE Доставка (

идентификатор\_доставки SERIAL PRIMARY KEY,

идентификатор\_подписки INT,

дата\_доставки DATE NOT NULL,

стоимость\_доставки DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (идентификатор\_подписки) REFERENCES Подписки(идентификатор\_подписки)

);

При помощи кода, написанного выше мы создаем 4 таблицы со следующей структурой.

Таблица «Издания»:

* индекс\_издания: Первичный ключ (PK).
* название\_издания, вид\_издания, цена\_1\_экземпляра: Атрибуты издания.
* Таблица «Получатели»:
* идентификатор\_получателя: Первичный ключ (PK), автоинкрементируемый.
* фио\_подписчика, улица, номер\_дома, номер\_квартиры: Атрибуты получателя.

Таблица «Подписки»:

* идентификатор\_подписки: Первичный ключ (PK), автоинкрементируемый.
* идентификатор\_получателя: Внешний ключ (FK), связывающий с таблицей Получатели.
* индекс\_издания: Внешний ключ (FK), связывающий с таблицей Издания.
* дата\_начала\_подписки, срок\_подписки, стоимость\_подписки: Атрибуты подписки.

Таблица «Доставка»:

* идентификатор\_доставки: Первичный ключ (PK), автоинкрементируемый.
* идентификатор\_подписки: Внешний ключ (FK), связывающий с таблицей Подписки.
* дата\_доставки, стоимость\_доставки: Атрибуты доставки.

Подключимся к базе данных через консоль и посмотрим на содержимое базы данных (Рисунок 2).

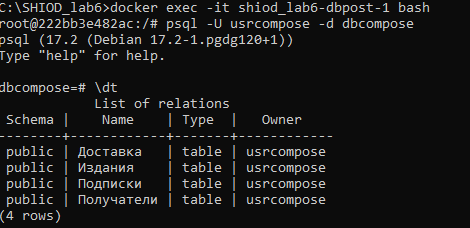


Рисунок 2 – Содержимое базы данных

В DBeaver посмотрим схему созданной базы данных (Рисунок 3).

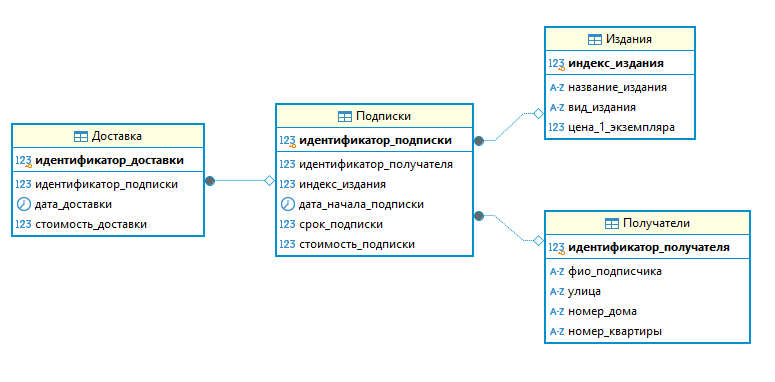


Рисунок 3 – Схема базы данных

Ссылка на репозиторий: https://github.com/IgorVlasov1/SHIOD\_LAB8

Ответы на контрольные вопросы:

1. Открытый исходный код: PostgreSQL является системой с открытым исходным кодом, что позволяет ее свободное использование и модификацию. Расширяемость: Поддерживает множество расширений и имеет возможность добавления собственных функций, типов данных, операторов и индексов. Соблюдение стандартов: PostgreSQL строго соблюдает стандарты SQL, что делает ее надежным выбором для многих приложений. Поддержка ACID-транзакций: Обеспечивает надежные транзакции с поддержкой всех свойств ACID (Atomicity, Consistency, Isolation, Durability). Репликация и отказоустойчивость: Поддерживает множество способов репликации данных, включая синхронную и асинхронную репликацию для обеспечения высокой доступности. Поддержка JSON и JSONB: Позволяет хранить и обрабатывать данные в формате JSON, что делает ее удобной для работы с документно-ориентированными структурами.
2. DDL (Data Definition Language) — это язык определения данных, используемый для описания структуры базы данных и объектов базы данных, таких как таблицы, индексы и схемы. Примеры команд DDL: CREATE: Создание объектов базы данных (таблиц, индексов, представлений и т.д.). ALTER: Изменение структуры существующих объектов базы данных. DROP: Удаление объектов базы данных.
3. DML (Data Manipulation Language) — это язык манипуляции данными, используемый для работы с данными в базе данных. Примеры команд DML: SELECT: Запрос данных из одной или нескольких таблиц. INSERT: Добавление новых записей в таблицу. UPDATE: Обновление существующих записей в таблице. DELETE: Удаление записей из таблицы.
4. DCL (Data Control Language) — это язык управления данными, используемый для управления доступом к данным в базе данных. Примеры команд DCL: GRANT: Предоставление пользователю определенных привилегий. REVOKE: Отзыв привилегий у пользователя.
5. Запросы начинаются с команды SELECT для извлечения данных из базы данных.
6. CREATE DATABASE ИмяБазыДанных.
7. Команда для создания таблицы в PostgreSQL: CREATE TABLE ИмяТаблицы (имя\_столбца тип\_данных ограничения).
8. SELECT \* FROM employee WHERE sal IN (1500, 3000).
9. SELECT \* FROM employee ORDER BY dept\_id NULLS FIRST.
10. Тип char в PostgreSQL предназначен для хранения фиксированной длины строк. Подходят следующие значения:’1’, ‘a’, ‘char’.
11. Select список полей from список таблиц union Select список полей from список таблиц order by поля для сортировки — НЕверно. ORDER BY должен применяться отдельно к каждому запросу перед оператором UNION. Select список полей from список таблиц having условия группировки group by поля для группировки — НЕверно. HAVING используется после GROUP BY, а не перед ним. Select список полей from список таблиц where условия group by поля для группировки order by поля для сортировки — Верно. Select список полей from список таблиц order by поля для сортировки union Select список полей from список таблиц order by поля для сортировки — НЕверно. Опять же, ORDER BY должен применяться отдельно к каждому запросу перед UNION.
12. Select avg (advance), sum (sales) from titles.