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-251: Ивченко Я.А.

подпись, дата

Воронеж 2025

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

**Ход выполнения работы:**

На основе логической модели была разработана физическая структура базы данных, состоящая из пяти взаимосвязанных таблиц. Для хранения информации о клиентах создана таблица Client с первичным ключом ClientID. Справочник отелей реализован в таблице Hotel с ключом HotelID. Таблица Tour содержит информацию о туристических предложениях, связывая отели с конкретными датами и ценами. Бронирования клиентов хранятся в таблице Booking, а финансовые операции - в таблице Payment.

Разработан SQL-скрипт создания базы данных с учетом всех ограничений целостности.

**-- Создание таблицы Клиентов**

**CREATE TABLE Client (**

**ClientID SERIAL PRIMARY KEY,**

**FullName VARCHAR(100) NOT NULL,**

**Phone VARCHAR(20) NOT NULL,**

**Email VARCHAR(100) UNIQUE NOT NULL, -- Ограничение UNIQUE гарантирует уникальность email**

**PassportNumber VARCHAR(50) UNIQUE NOT NULL -- UNIQUE гарантирует уникальность PassportNumber**

**);**

**-- Создание таблицы Отелей**

**CREATE TABLE Hotel (**

**HotelID SERIAL PRIMARY KEY,**

**HotelName VARCHAR(50) NOT NULL,**

**Stars INTEGER CHECK (Stars >= 1 AND Stars <= 5), -- Ограничение CHECK проверяет, что рейтинг от 1 до 5**

**Address TEXT NOT NULL -- Тип TEXT для длинных строк без явного ограничения длины**

**);**

**-- Создание таблицы Туров**

**CREATE TABLE Tour (**

**TourID SERIAL PRIMARY KEY,**

**HotelID INTEGER NOT NULL,**

**Country VARCHAR(100) NOT NULL,**

**StartDate DATE NOT NULL,**

**EndDate DATE NOT NULL,**

**Price NUMERIC(10, 2) NOT NULL CHECK (Price >= 0), -- Цена не может быть отрицательной**

**FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID) ON DELETE CASCADE -- ON DELETE CASCADE: при удалении жилья удалятся и связанные туры**

**);**

**-- Создание таблицы Бронировани**

**CREATE TABLE Booking (**

**BookingID SERIAL PRIMARY KEY,**

**ClientID INTEGER NOT NULL,**

**TourID INTEGER NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT CURRENT\_DATE, -- Значение по умолчанию - текущая дата**

**Status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (Status IN ('pending', 'confirmed', 'cancelled')),**

**FOREIGN KEY (ClientID) REFERENCES Client(ClientID) ON DELETE CASCADE,**

**FOREIGN KEY (TourID) REFERENCES Tour(TourID) ON DELETE CASCADE**

**);**

**--Создание таблицы Платежей**

**CREATE TABLE Payment (**

**PaymentID SERIAL PRIMARY KEY,**

**BookingID INTEGER NOT NULL UNIQUE, -- UNIQUE: на одно бронирование может быть только один платеж (упрощенная модель)**

**PaymentDate DATE NOT NULL DEFAULT CURRENT\_DATE,**

**Amount NUMERIC(10, 2) NOT NULL CHECK (Amount > 0),**

**PaymentMethod VARCHAR(50) NOT NULL,**

**FOREIGN KEY (BookingID) REFERENCES Booking(BookingID) ON DELETE CASCADE**

**);**

Для поля Status в таблице Booking применено ограничение CHECK, разрешающее только значения 'pending', 'confirmed' и 'cancelled', что соответствует бизнес-логике процесса бронирования. Все внешние ключи настроены с опцией ON DELETE CASCADE для автоматического удаления зависимых записей.

Контейнеризация решения

База данных обернута в Docker-контейнер с использованием docker-compose. Конфигурация включает официальный образ PostgreSQL, настройки аутентификации и том для сохранения данных. SQL-скрипты размещены в специальной директории для автоматического выполнения при первой инициализации контейнера.

На рисунке 1 показан процесс запуска Docker-контейнера.

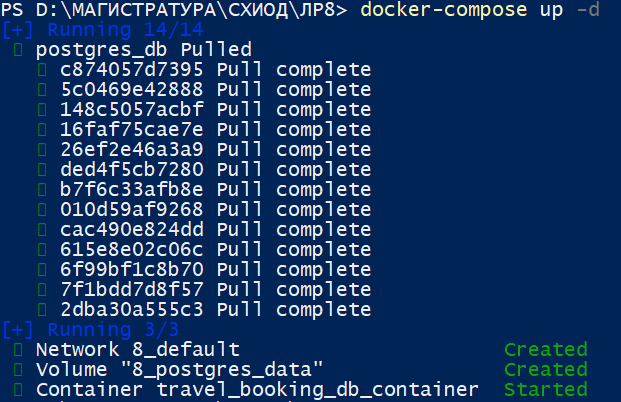


Рисунок 1 – Запуск Docker-контейнера

Средствами DBeaver построена графическая диаграмма, наглядно демонстрирующая поля и типы соответствующих таблиц, первичные и вторичные ключи и связи между сущностями. Диаграмма представлена на рисунке 2.

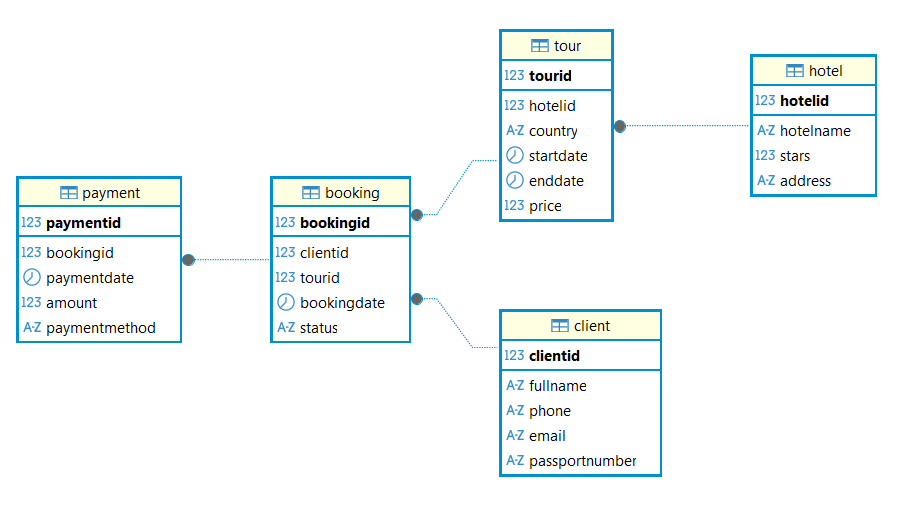


Рисунок 2 – Диаграмма физической структуры БД

**Вывод:** в результате лабораторной работы создана полностью функционирующая база данных, содержащая пять нормализованных таблиц с продуманными связями. Контейнеризация решения обеспечивает простоту развертывания и воспроизводимость среды. Диаграмма служит наглядным представлением структуры базы данных.

**Ответы на контрольные вопросы**

1. PostgreSQL — это объектно-реляционная СУБД с открытым исходным кодом. Ключевые особенности: поддержка сложных типов данных (JSON, XML, массивы), расширяемость через пользовательские функции и типы, строгая ACID-совместимость, мощная система индексов и полнотекстового поиска, а также развитые механизмы репликации и параллелизма.

2. DDL (Data Definition Language) — язык определения данных. Это набор команд для создания и изменения структуры базы данных: CREATE, ALTER, DROP. DDL используется для работы с метаданными — создания таблиц, индексов, ограничений, без манипуляций с самими данными.

3. DML (Data Manipulation Language) — язык манипулирования данными. Команды DML (SELECT, INSERT, UPDATE, DELETE) предназначены для работы с содержимым таблиц: выборка, добавление, изменение и удаление записей.

4. DCL (Data Control Language) — язык управления данными. Команды DCL (GRANT, REVOKE) управляют правами доступа к объектам базы данных, определяя, какие пользователи могут выполнять определенные операции.

5. Запрос обычно начинается с команды SELECT для выборки данных или с команд манипуляции (INSERT, UPDATE, DELETE). В PostgreSQL запрос может также начинаться с WITH для использования общих табличных выражений (CTE).

6. Для создания базы данных используется команда CREATE DATABASE.

7. Для создания таблицы применяется команда CREATE TABLE с указанием имени таблицы, столбцов и их типов данных.

8. Сотрудники с empno 7844, 7788 и 7902

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

9. Правильный ответ: SELECT \* FROM employee ORDER BY dept\_id NULLS FIRST;

По умолчанию NULL значения сортируются в конце. NULLS FIRST явно указывает разместить сотрудников без подразделения (dept\_id IS NULL) в начале списка.

10. Правильные ответы: '1', 'a', 'char'.

Тип char предназначен для символьных строк, поэтому допустимы только строковые значения в одинарных кавычках. Числа без кавычек и дробные числа не подходят.

11. Неверные конструкции:

Select список полей from список таблиц having условия группировки group by поля для группировки (неверный порядок: HAVING должен быть после GROUP BY)

Select список полей from список таблиц order by поля для сортировки union Select список полей from список таблиц order by поля для сортировки (нельзя использовать ORDER BY в каждой части UNION)

12. Правильный ответ: Select avg (advance), sum (sales) from titles group by type.

Задача требует вычислений по категориям, поэтому необходим GROUP BY type. Без группировки будут вычислены общие средние по всей таблице, а не по каждой категории отдельно.