Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение

высшего образования

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий

Кафедра «Прикладная информатика»

Форма обучения: очная

Лабораторная работа № 2

Тема: «Реализация структуры Data Vault»

по дисциплине

«Хранилища данных»

Студент В.М. Гребенникова

Москва 2025

# ВВЕДЕНИЕ

Третья нормальная форма (3NF) является обязательным этапом в процессе нормализации баз данных, который помогает организовать данные таким образом, чтобы минимизировать повторяемость и избежать аномалий при обновлении. Нормализация — это метод проектирования баз данных, который структурирует таблицы для всех конкретных данных и корректирует их. Применение 3NF позволяет создавать более устойчивую и управляемую структуру баз данных, что особенно важно для больших и сложных систем.

Основная идея Data Vault заключается в том, чтобы разместить данные вокруг бизнес-процессов и функций, а не отдельных предметных областей. Это позволяет более эффективно управлять данными и обеспечивать их целостность. Модель Data Vault сочетает в себе преимущества традиционных подходов, таких как третья нормальная форма (3NF), что делает ее гибридной.

Одним из главных преимуществ Data Vault является его гибкость. Модель позволяет легко вводить новые данные и изменять структуру без значительных затрат времени и ресурсов. Это делает Data Vault особенно подходящим для организаций, работающих с большими объемами данных и требующих быстрой адаптации к изменениям.

Вариант для выполнения лабораторной работы – 8. Система должна описывать порядок обработки заказов в службе быта.

# ЛАБОРАТОРНАЯ РАБОТА

Для создания базы данных, соответствующей требованиям третьей нормальной формы, были определены следующие необходимые таблицы:

* Таблица Клиенты – хранится основная информация о клиентах службы.
* Таблица Сотрудники – хранится основная информация о сотрудниках службы.
* Таблицы Услуги – хранится необходимая информация об услугах, предоставляемых службой быта.
* Таблица Заказы – хранится основная информация о заказах услуг.
* Таблица детали заказа – хранится дополнительная информация о заказах услуг.

Для того, чтобы создать базу данных выполним CREATE DATABASE everyday\_service (см. рисунок 1).

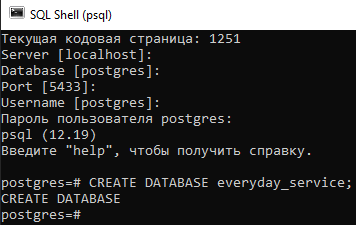


Рисунок 1 – Создание базы данных в PostgreSQL

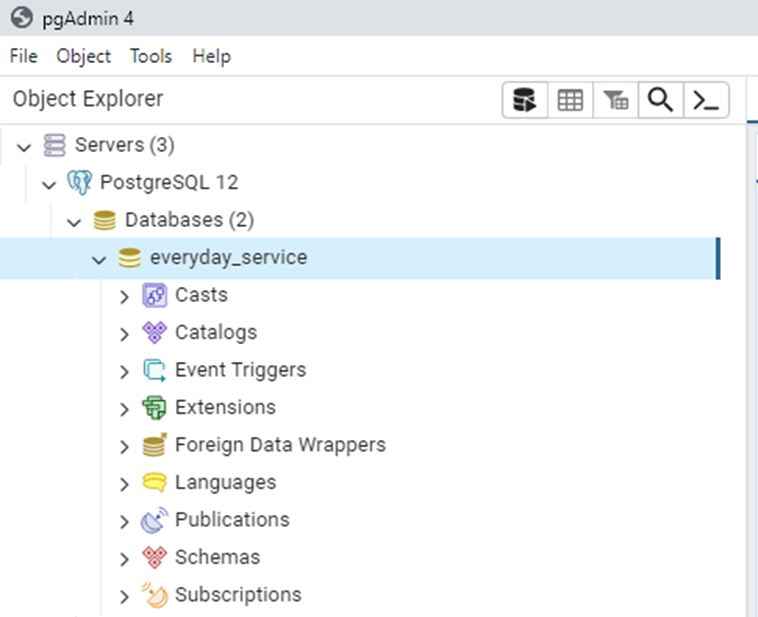


Рисунок 2 – Созданная БД

Для создания таблиц выполнима следующий sql-запрос в pgAdmin4:

*-- Таблица Клиенты*

CREATE TABLE Clients (

ClientID SERIAL PRIMARY KEY, *-- Уникальный идентификатор клиента*

Name VARCHAR(100) NOT NULL, *-- Имя клиента*

Phone VARCHAR(15), *-- Телефонный номер*

Email VARCHAR(100), *-- Электронная почта*

Address TEXT *-- Адрес клиента*

);

*-- Таблица Сотрудники*

CREATE TABLE Employees (

EmployeeID SERIAL PRIMARY KEY, *-- Уникальный идентификатор сотрудника*

Name VARCHAR(100) NOT NULL, *-- Имя сотрудника*

Role VARCHAR(50), *-- Должность*

Phone VARCHAR(15) *-- Телефонный номер*

);

*-- Таблица Услуги*

CREATE TABLE Services (

ServiceID SERIAL PRIMARY KEY, *-- Уникальный идентификатор услуги*

Name VARCHAR(100) NOT NULL, *-- Название услуги*

Description TEXT, *-- Описание услуги*

Price DECIMAL(10, 2) NOT NULL *-- Цена за единицу услуги*

);

*-- Таблица Заказы*

CREATE TABLE Orders (

OrderID SERIAL PRIMARY KEY, *-- Уникальный идентификатор заказа*

ClientID INT REFERENCES Clients(ClientID) ON DELETE CASCADE, *-- Клиент (FK)*

EmployeeID INT REFERENCES Employees(EmployeeID) ON DELETE SET NULL, *-- Сотрудник (FK)*

OrderDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата и время создания заказа*

Status VARCHAR(50) CHECK (Status IN ('Created', 'In Progress', 'Completed', 'Cancelled')), *-- Статус заказа*

TotalPrice DECIMAL(10, 2) *-- Итоговая стоимость заказа*

);

*-- Таблица Детали Заказа*

CREATE TABLE OrderDetails (

OrderDetailID SERIAL PRIMARY KEY, *-- Уникальный идентификатор записи*

OrderID INT REFERENCES Orders(OrderID) ON DELETE CASCADE, *-- Заказ (FK)*

ServiceID INT REFERENCES Services(ServiceID) ON DELETE CASCADE, *-- Услуга (FK)*

Quantity INT NOT NULL CHECK (Quantity > 0), *-- Количество единиц услуги*

SubTotal DECIMAL(10, 2) NOT NULL *-- Подытог по данной услуге*

);

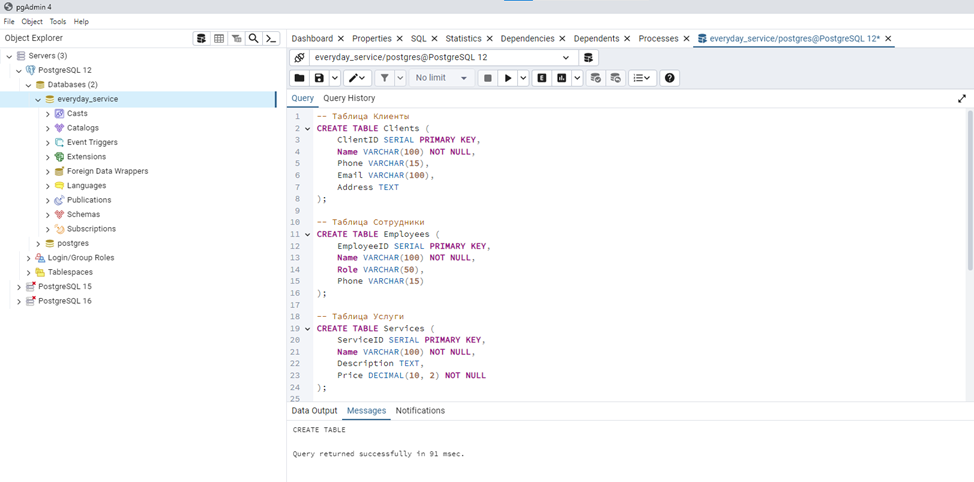


Рисунок 3 – Создание таблиц Клиенты, Сотрудники и Услуги

Выполним подключение и отображение списка всех таблиц (см. рисунок 4) в текущей базе данных: \c everyday\_service; \dt.

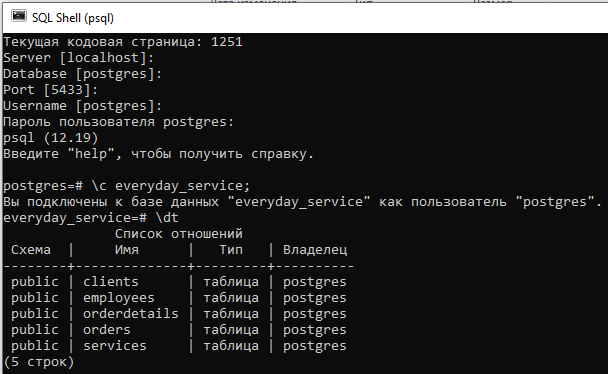


Рисунок 4 – Список таблиц

Перейдем к созданию базы данных в PostgreSQL Data Vault. Методология Data Vault предназначена для построения гибких, масштабируемых и адаптируемых хранилищ данных, особенно при работе с большими объемами данных и быстро меняющимися источниками информации. В отличие от традиционных моделей данных (например, Kimball или Inmon), Data Vault фокусируется на сохранении исторических данных и гибкости для интеграции новых источников данных без нарушения существующей архитектуры.

Основные элементы модели Data Vault:

Hubs (Хабы) — Хранят уникальные бизнес-ключи (например, идентификаторы клиентов, сотрудников и т.д.).

Links (Связи) — Хранят связи между различными хабами.

Satellites (Сателлиты) — Хранят детализированную информацию о хабах и связях, включая временные метки для историчности данных.

Перейдем к созданию. Хаб для Клиентов (hub\_clients). Хаб для клиентов будет хранить уникальные идентификаторы клиентов (в данном случае — ClientID).

CREATE TABLE hub\_clients (

client\_id SERIAL PRIMARY KEY, *-- Уникальный идентификатор клиента*

business\_key VARCHAR(100) NOT NULL, *-- Бизнес-ключ (например, уникальный идентификатор клиента)*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (business\_key) *-- Уникальный бизнес-ключ*

);

Хаб для Сотрудников (hub\_employees). Хаб для сотрудников будет хранить уникальные идентификаторы сотрудников (в данном случае — EmployeeID).

CREATE TABLE hub\_employees (

employee\_id SERIAL PRIMARY KEY, *-- Уникальный идентификатор сотрудника*

business\_key VARCHAR(100) NOT NULL, *-- Бизнес-ключ (например, уникальный идентификатор сотрудника)*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (business\_key) *-- Уникальный бизнес-ключ*

);

Хаб для Услуг (hub\_services). Хаб для услуг будет хранить уникальные идентификаторы услуг (в данном случае — ServiceID).

CREATE TABLE hub\_services (

service\_id SERIAL PRIMARY KEY, *-- Уникальный идентификатор услуги*

business\_key VARCHAR(100) NOT NULL, *-- Бизнес-ключ (например, уникальный идентификатор услуги)*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (business\_key) *-- Уникальный бизнес-ключ*

);

Links (Связи). Связь между Заказами и Клиентами (link\_orders\_clients). Эта связь будет представлять отношение между заказами и клиентами.

CREATE TABLE link\_orders\_clients (

order\_id INT REFERENCES hub\_orders(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб заказов*

client\_id INT REFERENCES hub\_clients(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб клиентов*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (order\_id, client\_id) *-- Уникальное сочетание заказ/клиент*

);

Связь между Заказами и Сотрудниками (link\_orders\_employees). Эта связь будет представлять отношение между заказами и сотрудниками.

CREATE TABLE link\_orders\_employees (

order\_id INT REFERENCES hub\_orders(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб заказов*

employee\_id INT REFERENCES hub\_employees(business\_key) ON DELETE SET NULL, *-- Внешний ключ на хаб сотрудников*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (order\_id, employee\_id) *-- Уникальное сочетание заказ/сотрудник*

);

Связь между Заказами и Услугами (link\_orders\_services). Эта связь будет представлять отношение между заказами и услугами.

CREATE TABLE link\_orders\_services (

order\_id INT REFERENCES hub\_orders(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб заказов*

service\_id INT REFERENCES hub\_services(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб услуг*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (order\_id, service\_id) *-- Уникальное сочетание заказ/услуга*

);

Satellites (Сателлиты). Сателлит для Клиентов (satellite\_clients). Этот сателлит будет содержать детализированную информацию о клиентах, такую как имя, телефон и адрес.

CREATE TABLE satellite\_clients (

client\_id INT REFERENCES hub\_clients(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб клиентов*

client\_name VARCHAR(100), *-- Имя клиента*

client\_phone VARCHAR(15), *-- Телефон клиента*

client\_email VARCHAR(100), *-- Электронная почта клиента*

client\_address TEXT, *-- Адрес клиента*

valid\_from TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Время начала действия записи*

valid\_to TIMESTAMP, *-- Время окончания действия записи*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (client\_id, valid\_from) *-- Уникальный ключ на основе времени*

);

Сателлит для Сотрудников (satellite\_employees). Этот сателлит будет содержать детализированную информацию о сотрудниках, такую как имя, телефон и роль.

CREATE TABLE satellite\_employees (

employee\_id INT REFERENCES hub\_employees(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб сотрудников*

employee\_name VARCHAR(100), *-- Имя сотрудника*

employee\_phone VARCHAR(15), *-- Телефон сотрудника*

employee\_role VARCHAR(50), *-- Должность сотрудника*

valid\_from TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Время начала действия записи*

valid\_to TIMESTAMP, *-- Время окончания действия записи*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (employee\_id, valid\_from) *-- Уникальный ключ на основе времени*

);

Сателлит для Услуг (satellite\_services). Этот сателлит будет содержать детализированную информацию о услугах, такую как описание и цена.

CREATE TABLE satellite\_services (

service\_id INT REFERENCES hub\_services(business\_key) ON DELETE CASCADE, *-- Внешний ключ на хаб услуг*

service\_description TEXT, *-- Описание услуги*

service\_price DECIMAL(10, 2), *-- Цена услуги*

valid\_from TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Время начала действия записи*

valid\_to TIMESTAMP, *-- Время окончания действия записи*

load\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, *-- Дата загрузки данных*

record\_source VARCHAR(50), *-- Источник данных*

PRIMARY KEY (service\_id, valid\_from) *-- Уникальный ключ на основе времени*

);

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была успешно создана база данных, соответствующая базовой форме (3NF), а также еще одна база данных, разработанная по методологии Data Vault.

Создание базы данных в 3NF позволяет минимизировать избыточность данных и избежать аномалий при обновлении, что значительно повышает целостность и управляемость информации. Нормализация данных принимает четкую структуру, где все неключевые атрибуты определяются только из начального ключа, что является основным требованием 3NF.

С другой стороны, применение модели Data Vault дало возможность создать гибкую и масштабируемую архитектуру хранилища данных. Эта модель, ориентированная на бизнес-процессы, позволяет легко адаптироваться к изменениям в требованиях и объемах данных, что особенно важно в условиях динамичного бизнеса. Хранение данных в виде видео хабов, сателлитов и ссылок обеспечивает высокий уровень подготовки и историчность данных.