Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Базы данных

Тема «Столовая Лидо»

Лабораторная работа №3

Реализация SQL-запросов для создания базы данных

Студент: М.С. Патюпин

Преподаватель: Д.В. Куприянова

МИНСК 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc190096954)

[1 СОЗДАНИЕ РЕАЛИЦИОННОЙ СХЕМЫ БАЗЫ ДАННЫХ 4](#_Toc190096955)

[1.1 Преобразование ER–объекта в реляционный объект 4](#_Toc190096956)

[1.2 Отображение связи «один-ко-многим» 6](#_Toc190096957)

[1.3 Отображение связи «многие-ко-многим» 8](#_Toc190096958)

[1.4 Реляционная схема базы данных 9](#_Toc190096959)

[2 СОЗДАНИЕ РЕАЛЯЦИОННОЙ МОДЕЛИ В pgAdmin 10](#_Toc190096960)

[2.1 Создание базы данных в pgAdmin 10](#_Toc190096961)

[2.2 Создание реляционной модели средствами pgAdmin 13](#_Toc190096962)

[ЗАКЛЮЧЕНИЕ 17](#_Toc190096963)

[ПРИЛОЖЕНИЕ А 18](#_Toc190096964)

# ВВЕДЕНИЕ

В лабораторной работе выполняется реализация схемы базы данных по ранее построенной реляционной схеме данных (см. лабораторную работу

№2). Требуется сформировать SQL-запросы для создания таблиц базы данных и выполнить их в СУБД. Требуется заполнить таблицы данными с помощью оператора INSERT.

Порядок выполнения работы:

1 Создать в СУБД новую схему данных для хранения пользовательских объектов (см. часть 2).

2 В этой новой схеме данных с помощью скрипта с запросами на языке DDL SQL реализовать таблицы, соответствующие реляционным отношениям схемы данных полученной в лабораторной работе №2.

3 Заполнить с помощью SQL-скрипта с использованием оператора INSERT таблицы строками данных для проверки правильного выбора первичных ключей и работоспособности ссылок между таблицами.

4 Рассмотреть простые действия по изменению структуры таблицы (переименование столбца таблицы, добавление и удаление ограничений на столбец таблицы или всю таблицу) и реализовать их с помощью оператора ALTER TABLE.

6 Создать временную таблицу с помощью оператора CREATE TABLE и удалить ее с помощью оператора DROP TABLE.

7 Экспортировать результаты работы в SQL-скрипт (см. часть 2), сравнить полученный скрипт со скриптами, созданными на этапах 2 и 3.

**1 СОЗДАНИЕ ТАБЛИЦ**

Для создания таблиц базы данных postgresql с помощью клиента pgAdmin необходимо:

1 Запустить клиент pgAdmin, рисунок 1.1.

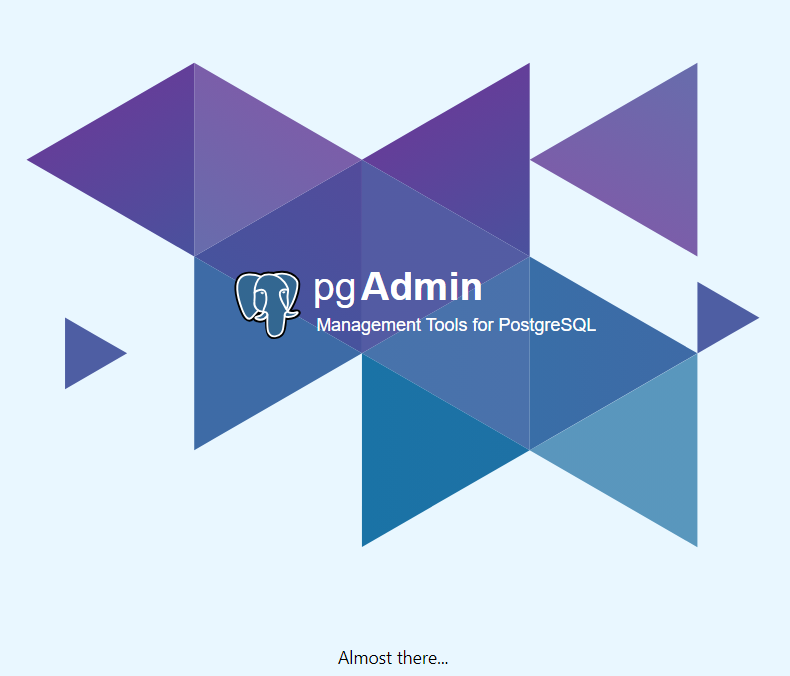


Рисунок 1.1 – Запуск клиента pgAdmin

2 Открыть окно исполнения команд, Obgect Explotet –> Servers –> PostgreSQL 16 –> Lido –> Query Tool, рисунок 1.2.

3 В появившейся вкладке, написать исходный код программы на языке программирования SQL для создания таблиц базы данных. Выполнить нажатием клавиши F5. Рисунок 1.3. Исходный код программы приведен в приложении А.

5 Ожидаем сообщения о создании новых таблиц, рисунок 1.4.

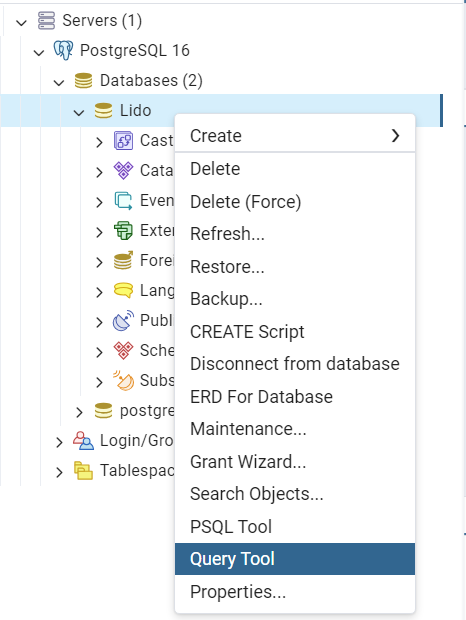


Рисунок 1.2 – Открытие Query Tool

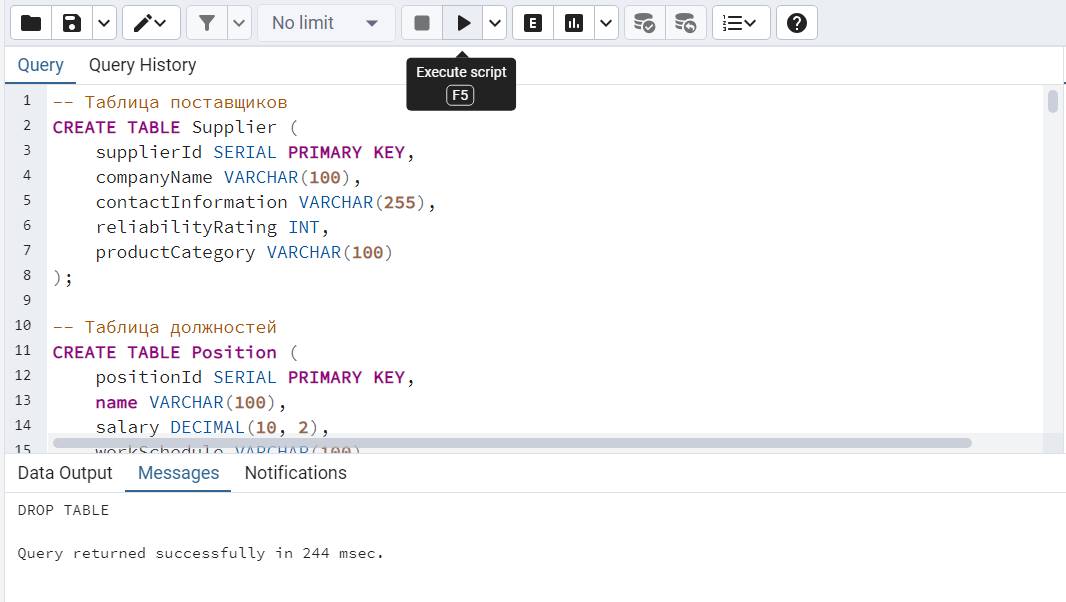


Рисунок 1.3 – Окно написания и для выполнения исходного кода программы

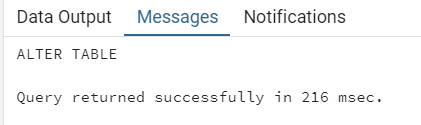


Рисунок 1.4 – Сообщение об успешном создании(обновлении) таблиц

1.2 Заполнение с помощью SQL-скрипта

Первыми для заполнения подлежат таблицы, не ссылающиеся на другие, или же мастер-таблицы:

– supplier: таблица поставщиков, приложение Б,

– position: таблица должностей,

– employee: таблица сотрудников,

– dish: таблица блюд,

– visitor: таблица посетителей.

# ЗАКЛЮЧЕНИЕ

В результате работы была преобразована ER-диаграмма в реляционную схему двумя способами:

– с помощью алгоритма, приведенного в разделе «Введение»;

– с помощью встроенных инструментов в программном обеспечении pgAdmin.

Для этого были определены и созданы таблицы в postrgresql.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Исходный код программы**

-- Таблица поставщиков

CREATE TABLE Supplier (

supplierId SERIAL PRIMARY KEY,

companyName VARCHAR(100),

contactInformation VARCHAR(255),

reliabilityRating INT,

productCategory VARCHAR(100)

);

-- Таблица должностей

CREATE TABLE Position (

positionId SERIAL PRIMARY KEY,

name VARCHAR(100),

salary DECIMAL(10, 2),

workSchedule VARCHAR(100),

responsibilities TEXT

);

-- Таблица сотрудников

CREATE TABLE Employee (

employeeId SERIAL PRIMARY KEY,

fullName VARCHAR(200),

phoneNumber VARCHAR(15),

email VARCHAR(100),

hireDate DATE,

contractEndDate DATE

);

-- Таблица блюд

CREATE TABLE Dish (

dishId SERIAL PRIMARY KEY,

name VARCHAR(100),

weight DECIMAL(10, 2),

nutritionalValue VARCHAR(100),

cost NUMERIC(10, 2),

category VARCHAR(100)

);

-- Таблица ингредиентов

CREATE TABLE Ingredient (

ingredientId SERIAL PRIMARY KEY,

name VARCHAR(100),

unitOfMeasurement VARCHAR(50),

inStock DECIMAL(10, 2),

expirationDate DATE,

supplierId INT

);

-- Таблица посетителей

CREATE TABLE Visitor (

visitorId SERIAL PRIMARY KEY,

name VARCHAR(200),

birthDate DATE,

phoneNumber VARCHAR(15),

preferences TEXT

);

-- Таблица заказов

CREATE TABLE "Order" (

orderId SERIAL PRIMARY KEY,

status VARCHAR(50),

totalCost DECIMAL(10, 2),

numberOfGuests INT,

orderDateTime TIMESTAMP,

visitorId INT,

employeeId INT

);

-- Таблица связей сотрудников и должностей

CREATE TABLE EmployeePosition (

employeeId INT,

positionId INT,

PRIMARY KEY (employeeId, positionId)

);

-- Таблица связей блюд и ингредиентов

CREATE TABLE DishIngredient (

dishId INT,

ingredientId INT,

PRIMARY KEY (dishId, ingredientId)

);

-- Таблица связей заказов и блюд

CREATE TABLE OrderDish (

orderId INT,

dishId INT,

PRIMARY KEY (orderId, dishId)

);

-- Добавление ограничений с помощью ALTER TABLE

ALTER TABLE Supplier

ALTER COLUMN companyName SET NOT NULL,

ADD CONSTRAINT companyName\_check CHECK (companyName ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ADD CONSTRAINT reliabilityRating\_check CHECK (reliabilityRating BETWEEN 1 AND 10);

ALTER TABLE Position

ALTER COLUMN name SET NOT NULL,

ADD CONSTRAINT name\_check CHECK (name ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ALTER COLUMN salary SET NOT NULL,

ADD CONSTRAINT salary\_check CHECK (salary >= 0);

ALTER TABLE Employee

ALTER COLUMN fullName SET NOT NULL,

ADD CONSTRAINT fullName\_check CHECK (fullName ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ADD CONSTRAINT phoneNumber\_check CHECK (phoneNumber ~ '^\+375[0-9]{9}$'),

ADD CONSTRAINT email\_check CHECK (email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),

ADD CONSTRAINT contractEndDate\_check CHECK (contractEndDate >= hireDate);

ALTER TABLE Dish

ALTER COLUMN name SET NOT NULL,

ADD CONSTRAINT name\_check CHECK (name ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ALTER COLUMN weight SET NOT NULL,

ADD CONSTRAINT weight\_check CHECK (weight > 0),

ADD CONSTRAINT cost\_check CHECK (cost > 0);

ALTER TABLE Ingredient

ALTER COLUMN name SET NOT NULL,

ADD CONSTRAINT name\_check CHECK (name ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ALTER COLUMN inStock SET NOT NULL,

ADD FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId);

ALTER TABLE Visitor

ALTER COLUMN name SET NOT NULL,

ADD CONSTRAINT name\_check CHECK (name ~ '^[a-zA-Zа-яА-Я0-9 ,.-]+$'),

ADD CONSTRAINT phoneNumber\_check CHECK (phoneNumber ~ '^\+375[0-9]{9}$');

ALTER TABLE "Order"

ALTER COLUMN totalCost SET NOT NULL,

ADD CONSTRAINT totalCost\_check CHECK (totalCost >= 0),

ADD CONSTRAINT numberOfGuests\_check CHECK (numberOfGuests >= 0),

ALTER COLUMN orderDateTime SET NOT NULL,

ADD FOREIGN KEY (visitorId) REFERENCES Visitor(visitorId),

ADD FOREIGN KEY (employeeId) REFERENCES Employee(employeeId);

ALTER TABLE EmployeePosition

ADD FOREIGN KEY (employeeId) REFERENCES Employee(employeeId),

ADD FOREIGN KEY (positionId) REFERENCES Position(positionId);

ALTER TABLE DishIngredient

ADD FOREIGN KEY (dishId) REFERENCES Dish(dishId),

ADD FOREIGN KEY (ingredientId) REFERENCES Ingredient(ingredientId);

ALTER TABLE OrderDish

ADD FOREIGN KEY (orderId) REFERENCES "Order"(orderId),

ADD FOREIGN KEY (dishId) REFERENCES Dish(dishId);

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Исходный код программы**