# Титульный лист

Предмет: Теория баз данных

Практическая работа №5

Название работы: Создание и Нормализация Базы Данных для Бизнес-Процесса Доставки

Группа: БИСО-02-23

ФИО: Макаревич Сергей Витальевич

Дата: [Текущая дата]

## Отчёт по Практической Работе

### Введение

Целью данной работы является выполнение задач, связанных с созданием базы данных, написанием SQL-запросов и анализом результатов соединений таблиц. В процессе выполнены задания, включающие создание таблиц, внесение данных, выполнение различных типов JOIN-запросов, а также использование вложенных запросов.

### 1. ER-Диаграмма

Выбраны три таблицы:

1. **Customer** — содержит информацию о клиентах.
2. **Employee** — содержит данные о сотрудниках.
3. **Delivery** — хранит данные о заказах, связывая клиентов и сотрудников.

Связи между таблицами:

* Таблица Delivery связана с Customer через внешний ключ customer\_id.
* Таблица Delivery связана с Employee через внешний ключ employee\_id.

**Добавьте сюда скриншот ER-диаграммы.**

### 2. Создание таблиц

**SQL-команды для создания таблиц:**

-- Таблица Customer

CREATE TABLE Customer (

customer\_id SERIAL PRIMARY KEY,

customer\_name VARCHAR(100) NOT NULL,

customer\_phone VARCHAR(20),

customer\_address VARCHAR(255)

);

-- Таблица Employee

CREATE TABLE Employee (

employee\_id SERIAL PRIMARY KEY,

employee\_name VARCHAR(100) NOT NULL,

employee\_position VARCHAR(50)

);

-- Таблица Delivery

CREATE TABLE Delivery (

order\_id SERIAL PRIMARY KEY,

customer\_id INT REFERENCES Customer(customer\_id) ON DELETE CASCADE ON UPDATE CASCADE,

employee\_id INT REFERENCES Employee(employee\_id) ON DELETE CASCADE ON UPDATE CASCADE,

item\_name VARCHAR(100),

item\_weight FLOAT,

item\_volume FLOAT,

item\_quantity INT,

origin\_address VARCHAR(255),

destination\_address VARCHAR(255),

shipment\_date DATE,

delivery\_date DATE,

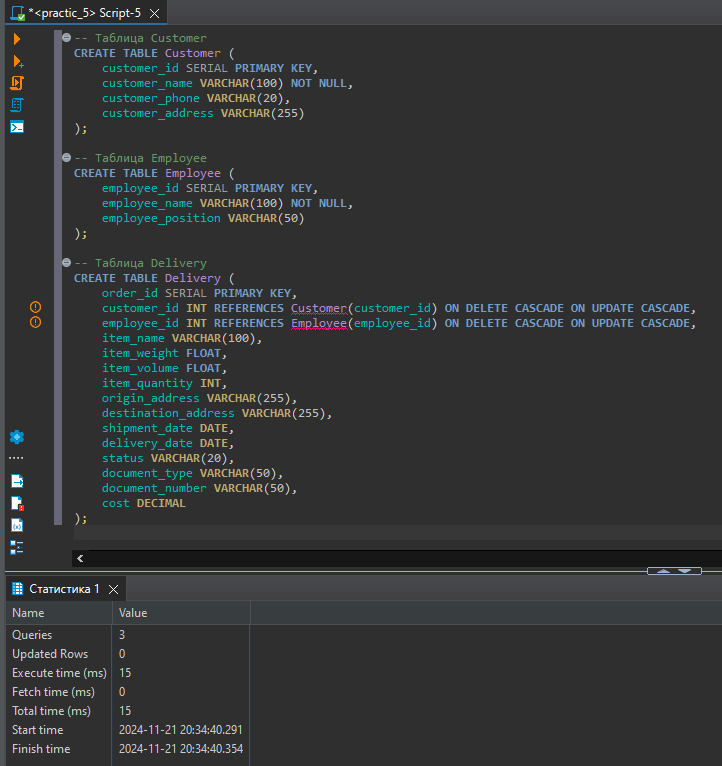
status VARCHAR(20),

document\_type VARCHAR(50),

document\_number VARCHAR(50),

cost DECIMAL

);



**Обоснование последовательности создания таблиц:**

1. Сначала создаются родительские таблицы Customer и Employee, так как они не зависят от других данных.
2. Затем создаётся таблица Delivery, ссылающаяся на записи из родительских таблиц.

### 3. Внесение данных

**SQL-команды для вставки данных:**

-- Вставка данных в Customer

INSERT INTO Customer (customer\_name, customer\_phone, customer\_address) VALUES

('Ivan Ivanov', '1234567890', 'Address 1'),

('Petr Petrov', '0987654321', 'Address 2'),

('Anna Smirnova', '1122334455', 'Address 3'),

('Maria Ivanova', '5566778899', 'Address 4'),

('Nikolay Sidorov', '6677889900', 'Address 5');

-- Вставка данных в Employee

INSERT INTO Employee (employee\_name, employee\_position) VALUES

('John Doe', 'Driver'),

('Jane Smith', 'Dispatcher'),

('Alex Black', 'Manager'),

('Olga White', 'Loader'),

('Victor Green', 'Driver');

-- Вставка данных в Delivery

INSERT INTO Delivery (customer\_id, employee\_id, item\_name, item\_weight, item\_volume, item\_quantity, origin\_address, destination\_address, shipment\_date, delivery\_date, status, document\_type, document\_number, cost) VALUES

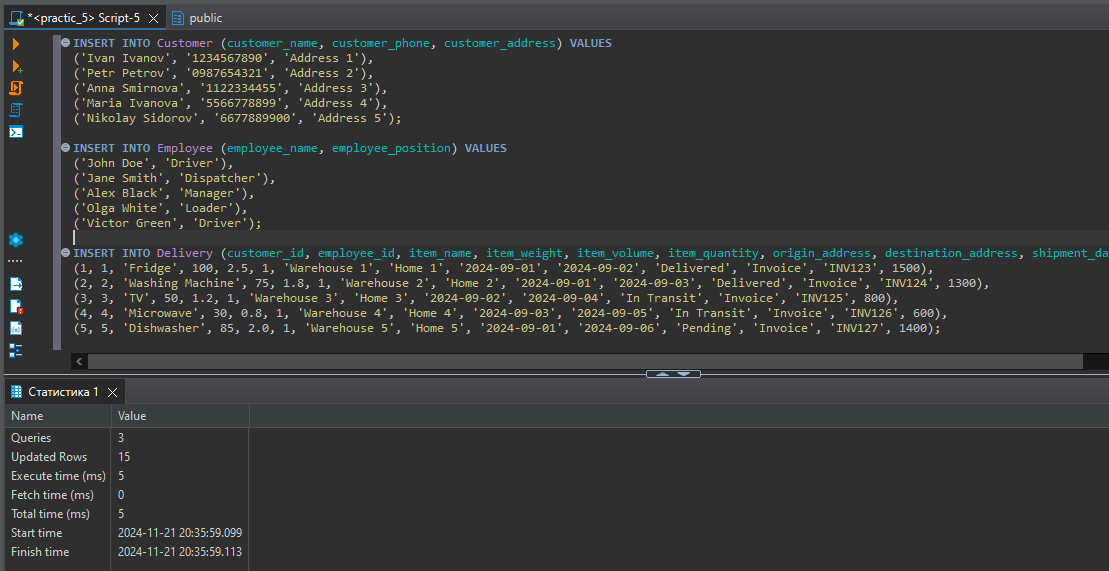
(1, 1, 'Fridge', 100, 2.5, 1, 'Warehouse 1', 'Home 1', '2024-09-01', '2024-09-02', 'Delivered', 'Invoice', 'INV123', 1500),

(2, 2, 'Washing Machine', 75, 1.8, 1, 'Warehouse 2', 'Home 2', '2024-09-01', '2024-09-03', 'Delivered', 'Invoice', 'INV124', 1300),

(3, 3, 'TV', 50, 1.2, 1, 'Warehouse 3', 'Home 3', '2024-09-02', '2024-09-04', 'In Transit', 'Invoice', 'INV125', 800),

(4, 4, 'Microwave', 30, 0.8, 1, 'Warehouse 4', 'Home 4', '2024-09-03', '2024-09-05', 'In Transit', 'Invoice', 'INV126', 600),

(5, 5, 'Dishwasher', 85, 2.0, 1, 'Warehouse 5', 'Home 5', '2024-09-01', '2024-09-06', 'Pending', 'Invoice', 'INV127', 1400);



**Обоснование последовательности внесения данных:**

* Сначала заполняются таблицы Customer и Employee, чтобы можно было ссылаться на их ключи в таблице Delivery.
* Если попытаться вставить запись в таблицу Delivery, не добавив соответствующие записи в Customer или Employee, произойдёт ошибка из-за нарушения ссылочной целостности.

### 4. INNER JOIN

**SQL-запрос:**

SELECT Delivery.order\_id, Customer.customer\_name, Employee.employee\_name, Delivery.item\_name, Delivery.status

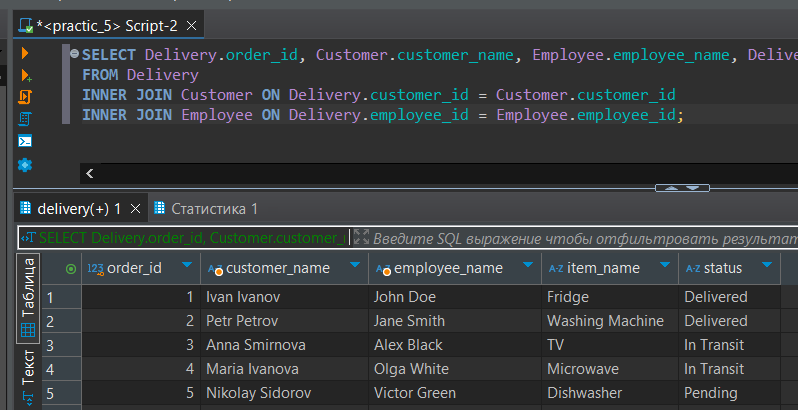
FROM Delivery

INNER JOIN Customer ON Delivery.customer\_id = Customer.customer\_id

INNER JOIN Employee ON Delivery.employee\_id = Employee.employee\_id;

**Результат:**

* Выводятся только те записи, у которых есть соответствия во всех трёх таблицах.
* Записи без соответствий исключаются.



### 5. LEFT OUTER JOIN

**SQL-запрос:**

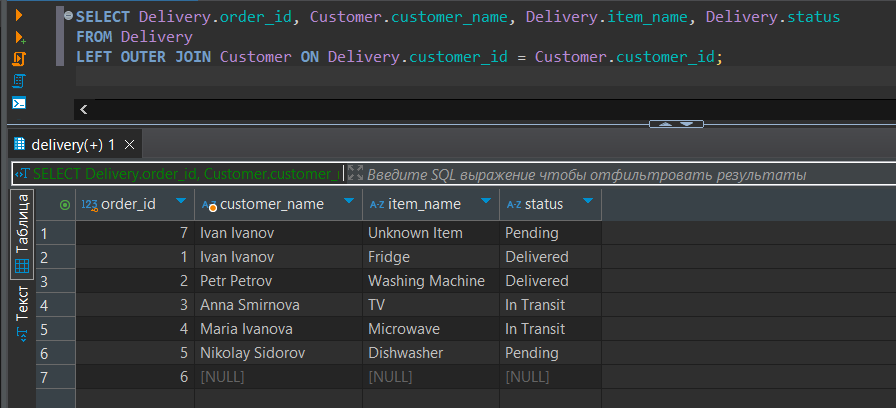
SELECT Delivery.order\_id, Customer.customer\_name, Delivery.item\_name, Delivery.status

FROM Delivery

LEFT OUTER JOIN Customer ON Delivery.customer\_id = Customer.customer\_id;

**Результат:**

* Все записи из таблицы Delivery с соответствующими данными из Customer. Для записей без соответствий поля из Customer заполнены NULL.



### 6. RIGHT OUTER JOIN

**SQL-запрос:**

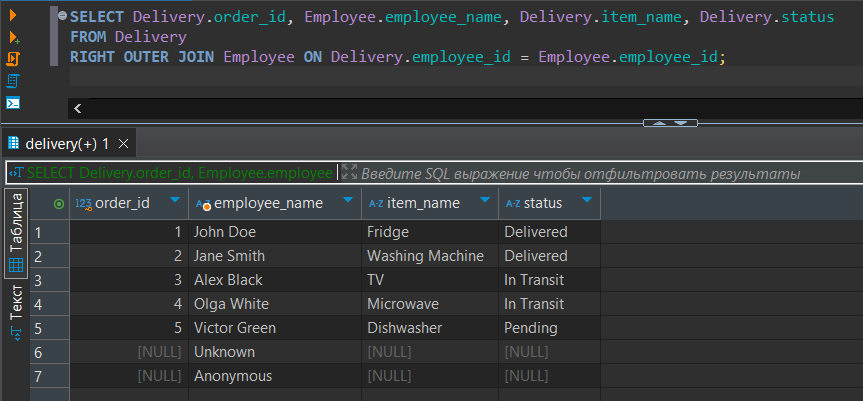
SELECT Delivery.order\_id, Employee.employee\_name, Delivery.item\_name, Delivery.status

FROM Delivery

RIGHT OUTER JOIN Employee ON Delivery.employee\_id = Employee.employee\_id;

**Результат:**

* Все записи из таблицы Employee с соответствующими данными из Delivery. Для записей без соответствий поля из Delivery заполнены NULL.



### 7. RIGHT JOIN (перестановка таблиц)

**SQL-запрос:**

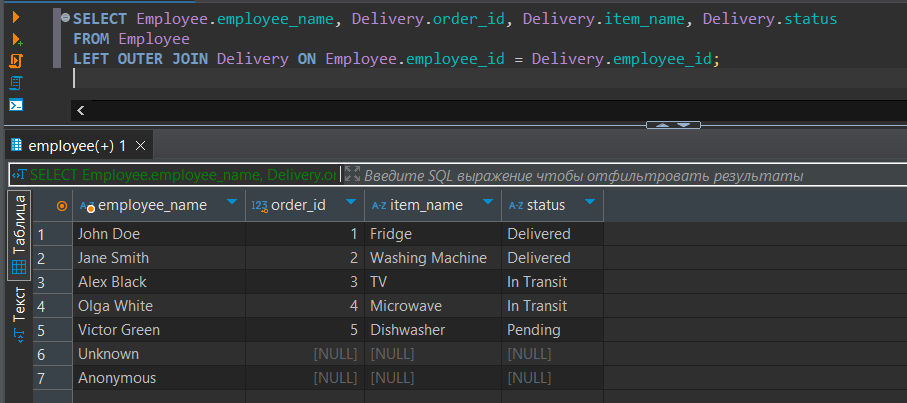
SELECT Employee.employee\_name, Delivery.order\_id, Delivery.item\_name, Delivery.status

FROM Employee

LEFT OUTER JOIN Delivery ON Employee.employee\_id = Delivery.employee\_id;

**Результат:**

* Результат аналогичен пункту 6, так как RIGHT JOIN с таблицами в обратном порядке эквивалентен LEFT JOIN.



### 8. FULL OUTER JOIN

**SQL-запрос:**

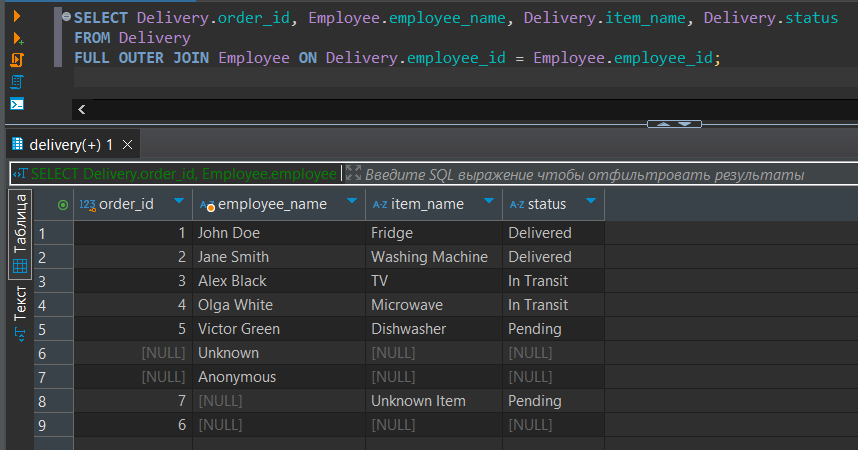
SELECT Delivery.order\_id, Employee.employee\_name, Delivery.item\_name, Delivery.status

FROM Delivery

FULL OUTER JOIN Employee ON Delivery.employee\_id = Employee.employee\_id;

**Результат:**

* Все записи из обеих таблиц. Поля без соответствий заполнены NULL.



### 9. NOT IN

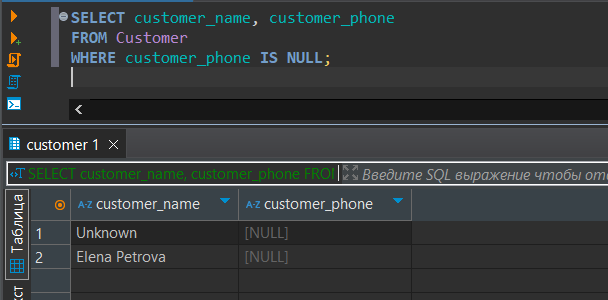
**SQL-запрос:**

SELECT customer\_name, customer\_phone

FROM Customer

WHERE customer\_id NOT IN (SELECT customer\_id FROM Delivery WHERE customer\_id IS NOT NULL);  
**Результат:**

* Клиенты, которые не имеют заказов.



### 10. EXISTS

**SQL-запрос:**

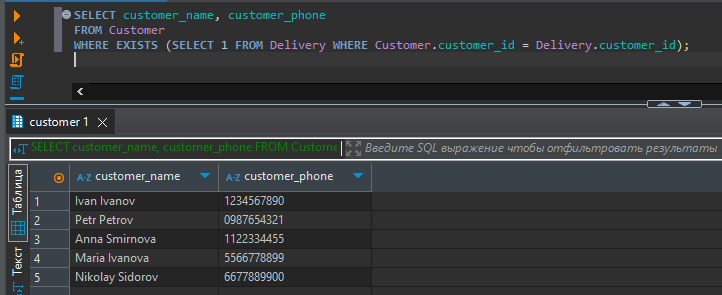
SELECT customer\_name, customer\_phone

FROM Customer

WHERE EXISTS (SELECT 1 FROM Delivery WHERE Customer.customer\_id = Delivery.customer\_id);

**Результат:**

* Клиенты, которые имеют хотя бы один заказ.

****