Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ POSTGRESQL

Отчет по лабораторной работе по дисциплине «Основы разработки баз данных»

Студент гр. 573-3

\_\_\_\_\_\_\_\_\_\_\_\_ Р.В. Слиньков

дата

\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Руководитель:

Преподаватель:

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Р. О. Остапенко

подпись

оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Томск, 2025

**Введение**

В современном мире базы данных играют ключевую роль в системах хранения и обработки информации. Они используются в самых разных областях — от небольших веб-приложений до крупных корпоративных систем. Эффективная организация данных позволяет оптимизировать работу программных продуктов, обеспечить целостность и безопасность информации, а также упростить её поиск и анализ.

Цель данной лабораторной работы — познакомиться с основными принципами проектирования реляционных баз данных, освоить синтаксис языка структурированных запросов SQL и научиться создавать таблицы, определять связи между ними, а также накладывать ограничения для обеспечения корректности данных. В процессе выполнения работы будет спроектирована и реализована структура базы данных на языке SQL в среде PostgreSQL, отражающая реальные сущности и их взаимосвязи.

**Задачи:**

Реализовать БД согласно требованиям:

1. БД должна быть реляционной;
2. БД должна быть нормализована по 3НФ (иди БКНФ);
3. БД должна состоять минимум из 3-4 связанных таблиц (на каждого студента); в таблицах БД должны быть наложены ограничения на поля таблиц (по усмотрению разработчиков решается, на какие поля будут наложены ограничения). Должно быть не менее 3 ограничений на разные поля таблиц (без учета ограничений PRIMARY KEY, FOREIGN KEY). Ограничения проверки, значения по умолчанию обязательно;
4. должна быть обеспечена ссылочная целостность;
5. В БД должны быть обязательно поля разных типов данных, в том числе поля:
   1. Позволяющие хранить рисунки;
   2. Позволяющие хранить длинный текст;
   3. Значения, в которых должны соответствовать списку значений, например Пол мужской или женский;
   4. Значения логического типа.
6. **ОСНОВНАЯ ЧАСТЬ**

**1.1 Описание данных для реализации БД**

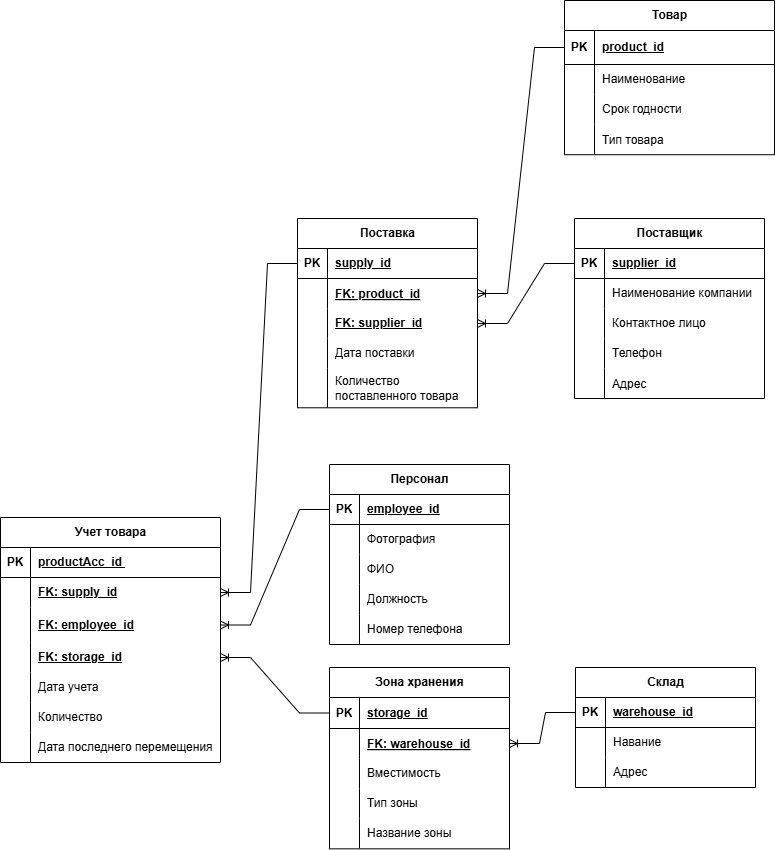
В ходе индивидуальных заданий была разработана реляционная модель данных (рисунок 1.1), которую необходимо реализовать в виде БД.

Рисунок 1.1 – Реляционная модель данных

Опишем сущности в виде таблиц, чтобы потом перенести их в БД. Также при необходимости добавим поля и ограничения, чтобы они соответствовали формулировке задания.

Для «Склад» (таблица 1.1) были описаны характеристики.

Таблица 1.1 – Описание «Склад»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничение |
| warehouse\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| name | VARCHAR(100) | Название склада. | Размер поля 100, NOT NULL. |
| address | VARCHAR(200) | Адрес склада. | Размер поля 200, NOT NULL. |

Для «Зона хранения» (таблица 1.2) было описаны характеристики, а также добавлено поле для хранения типа зоны хранения в виде дискретного значения.

Таблица 1.2 – Описание «Зона хранения»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| storage\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| warehouse\_id | INT | Идентификатор склада. | FK, NOT NULL. |
| capacity | INT | Вместимость зоны. | >= 0, NOT NULL. |
| zone\_type | zone\_type | Тип зоны (дискретный). | NOT NULL. |
| zone\_name | VARCHAR(100) | Название зоны. | Размер поля 100, NOT NULL. |

Для «Товар» (таблица 1.3) были описаны характеристики и добавлено поле для хранения возможных типов товара в виде дискретного значения.

Таблица 1.3 – Описание «Товар»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| product\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| name | VARCHAR(100) | Наименование товара. | Размер поля 100, NOT NULL. |
| expiry\_date | DATE | Срок годности. |  |
| product\_type | product\_type | Тип товара (дискретный). | NOT NULL. |
| is\_active | BOOLEAN | Активен ли товар. | NOT NULL, DEFAULT TRUE. |
| photo | BYTEA | Фотография товара. | Никаких ограничений не накладывается. |

Для «Поставщик» (таблица 1.4) были описаны характеристики.

Таблица 1.4 – Описание «Поставщик»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| supplier\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| company\_name | VARCHAR(100) | Название компании. | Размер поля 100, NOT NULL. |
| contact\_person | VARCHAR(100) | Контактное лицо. | Размер поля 100. |
| phone | VARCHAR(20) | Телефон. | Размер поля 20. |
| address | VARCHAR(200) | Адрес компании. | Размер поля 200. |

Для «Персонал» (таблица 1.5) были описаны характеристики.

Таблица 1.5 – Описание «Персонал»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| employee\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| photo | BYTEA | Фотография сотрудника. | Никаких ограничений не накладывается. |
| full\_name | VARCHAR(100) | ФИО сотрудника. | Размер поля 100, NOT NULL. |
| position | VARCHAR(50) | Должность. | Размер поля 50, NOT NULL. |
| phone | VARCHAR(20) | Номер телефона. | Размер поля 20. |

Для «Учет товара» (таблица 1.6) были описаны характеристики.

Таблица 1.6 – Описание «Учет товара»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| productAcc\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| supply\_id | INT | Идентификатор поставки. | FK, NOT NULL. |
| employee\_id | INT | Идентификатор сотрудника. | FK, NOT NULL. |
| storage\_id | INT | Идентификатор зоны хранения. | FK, NOT NULL. |
| accounting\_date | DATE | Дата учета. | NOT NULL. |
| quantity | INT | Количество товара. | >= 0, NOT NULL. |
| last\_movement\_date | DATE | Дата последнего перемещения. | Никаких ограничений не накладывается. |

Также необходимо описать таблицу «Поставка» (таблица 1.8) для хранения связи М:М.

Таблица 1.8 – Описание «Поставка»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Описание | Ограничения |
| supply\_id | SERIAL | Уникальный идентификатор. | PK, > 0. |
| product\_id | INT | Идентификатор товара. | FK, NOT NULL. |
| supplier\_id | INT | Идентификатор поставщика. | FK, NOT NULL. |
| supply\_date | DATE | Дата поставки. | NOT NULL. |
| quantity | INT | Количество товара. | > 0, NOT NULL. |

**1.2 Создание БД**

Для создания БД мы будем использовать систему управления БД (СУБД) PostgreSQL и pgAdmin. Она позволяет создавать таблицы как с помощью пользовательского кода, так и SQL кода и пользовательского интерфейса. Для нашего случая будем использовать SQL запросы и пользовательский интерфейс.

Начнём создание БД с её объявления (рисунок 1.2 и рисунок 1.3).

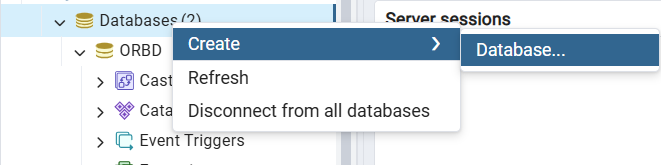


Рисунок 1.2 – Создание БД для склада

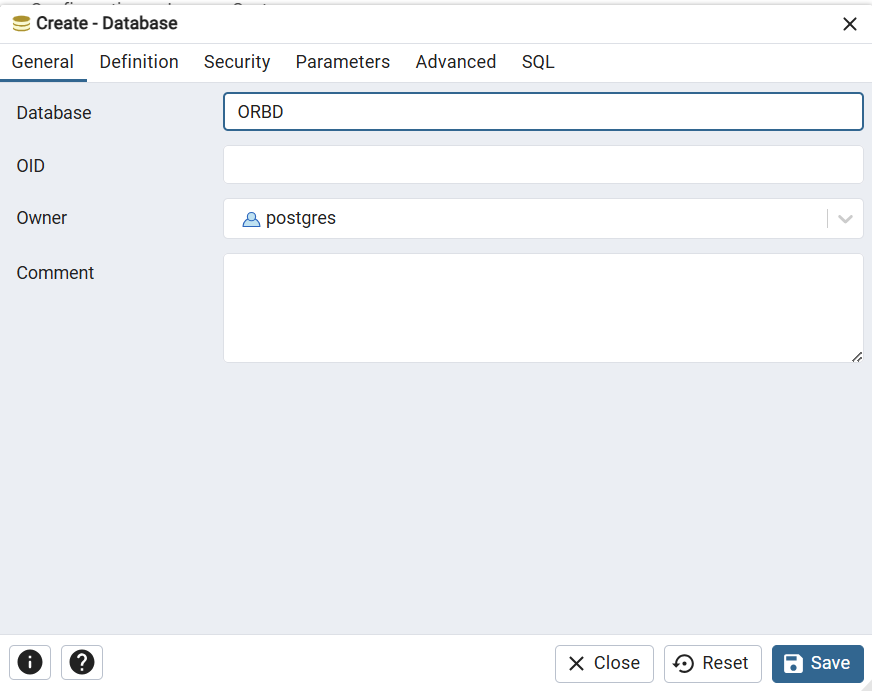


Рисунок 1.3 – Создание БД для склада

Как мы видим, наша БД отобразилась в интерфейсе pgAdmin (рисунок 1.3), и мы можем начинать работу с ней.

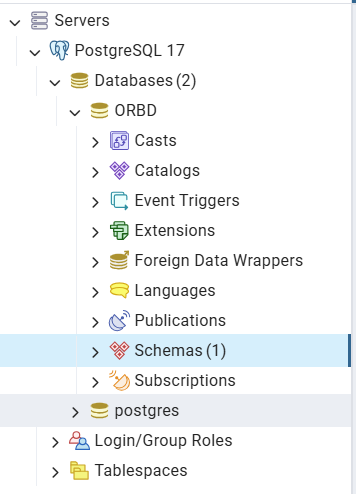


Рисунок 1.3 – Отображение БД в интерфейсе

Теперь напишем код для создания таблиц в нашей БД и связей между ними.

Листинг 1.1 – Создание таблиц

-- Создание типа ENUM для дискретных значений

CREATE TYPE product\_type AS ENUM ('food', 'electronics', 'clothing', 'other');

CREATE TYPE zone\_type AS ENUM ('refrigerated', 'dry', 'frozen', 'general');

-- Таблица "Склад"

CREATE TABLE warehouse (

warehouse\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

address VARCHAR(200) NOT NULL,

CONSTRAINT warehouse\_id\_positive CHECK (warehouse\_id > 0)

);

-- Таблица "Зона хранения"

CREATE TABLE storage\_zone (

storage\_id SERIAL PRIMARY KEY,

warehouse\_id INT NOT NULL,

capacity INT NOT NULL CHECK (capacity >= 0),

zone\_type zone\_type NOT NULL,

zone\_name VARCHAR(100) NOT NULL,

CONSTRAINT storage\_id\_positive CHECK (storage\_id > 0),

CONSTRAINT fk\_warehouse FOREIGN KEY (warehouse\_id) REFERENCES warehouse(warehouse\_id) ON DELETE RESTRICT

);

-- Таблица "Товар"

CREATE TABLE product (

product\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

expiry\_date DATE,

product\_type product\_type NOT NULL,

is\_active BOOLEAN NOT NULL DEFAULT TRUE,

photo BYTEA,

CONSTRAINT product\_id\_positive CHECK (product\_id > 0)

);

-- Таблица "Поставщик"

CREATE TABLE supplier (

supplier\_id SERIAL PRIMARY KEY,

company\_name VARCHAR(100) NOT NULL,

contact\_person VARCHAR(100),

phone VARCHAR(20),

address VARCHAR(200),

CONSTRAINT supplier\_id\_positive CHECK (supplier\_id > 0)

);

-- Таблица "Персонал"

CREATE TABLE employee (

employee\_id SERIAL PRIMARY KEY,

photo BYTEA,

full\_name VARCHAR(100) NOT NULL,

position VARCHAR(50) NOT NULL,

phone VARCHAR(20),

CONSTRAINT employee\_id\_positive CHECK (employee\_id > 0)

);

-- Таблица "Поставка"

CREATE TABLE supply (

supply\_id SERIAL PRIMARY KEY,

product\_id INT NOT NULL,

supplier\_id INT NOT NULL,

supply\_date DATE NOT NULL,

quantity INT NOT NULL CHECK (quantity > 0),

CONSTRAINT supply\_id\_positive CHECK (supply\_id > 0),

CONSTRAINT fk\_product FOREIGN KEY (product\_id) REFERENCES product(product\_id) ON DELETE RESTRICT,

CONSTRAINT fk\_supplier FOREIGN KEY (supplier\_id) REFERENCES supplier(supplier\_id) ON DELETE RESTRICT

);

-- Таблица "Учет товара"

CREATE TABLE product\_accounting (

productAcc\_id SERIAL PRIMARY KEY,

supply\_id INT NOT NULL,

employee\_id INT NOT NULL,

storage\_id INT NOT NULL,

accounting\_date DATE NOT NULL,

quantity INT NOT NULL CHECK (quantity >= 0),

last\_movement\_date DATE,

CONSTRAINT productAcc\_id\_positive CHECK (productAcc\_id > 0),

CONSTRAINT fk\_supply FOREIGN KEY (supply\_id) REFERENCES supply(supply\_id) ON DELETE RESTRICT,

CONSTRAINT fk\_employee FOREIGN KEY (employee\_id) REFERENCES employee(employee\_id) ON DELETE RESTRICT,

CONSTRAINT fk\_storage FOREIGN KEY (storage\_id) REFERENCES storage\_zone(storage\_id) ON DELETE RESTRICT

);

Теперь мы можем увидеть отображение всех таблиц в интерфейсе (рисунок 1.4) и на ERD схеме данных (рисунок 1.5).

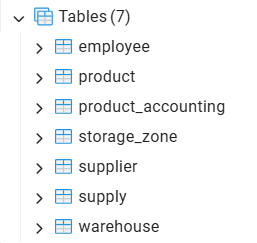


Рисунок 1.4 – Отображение таблиц в интерфейсе

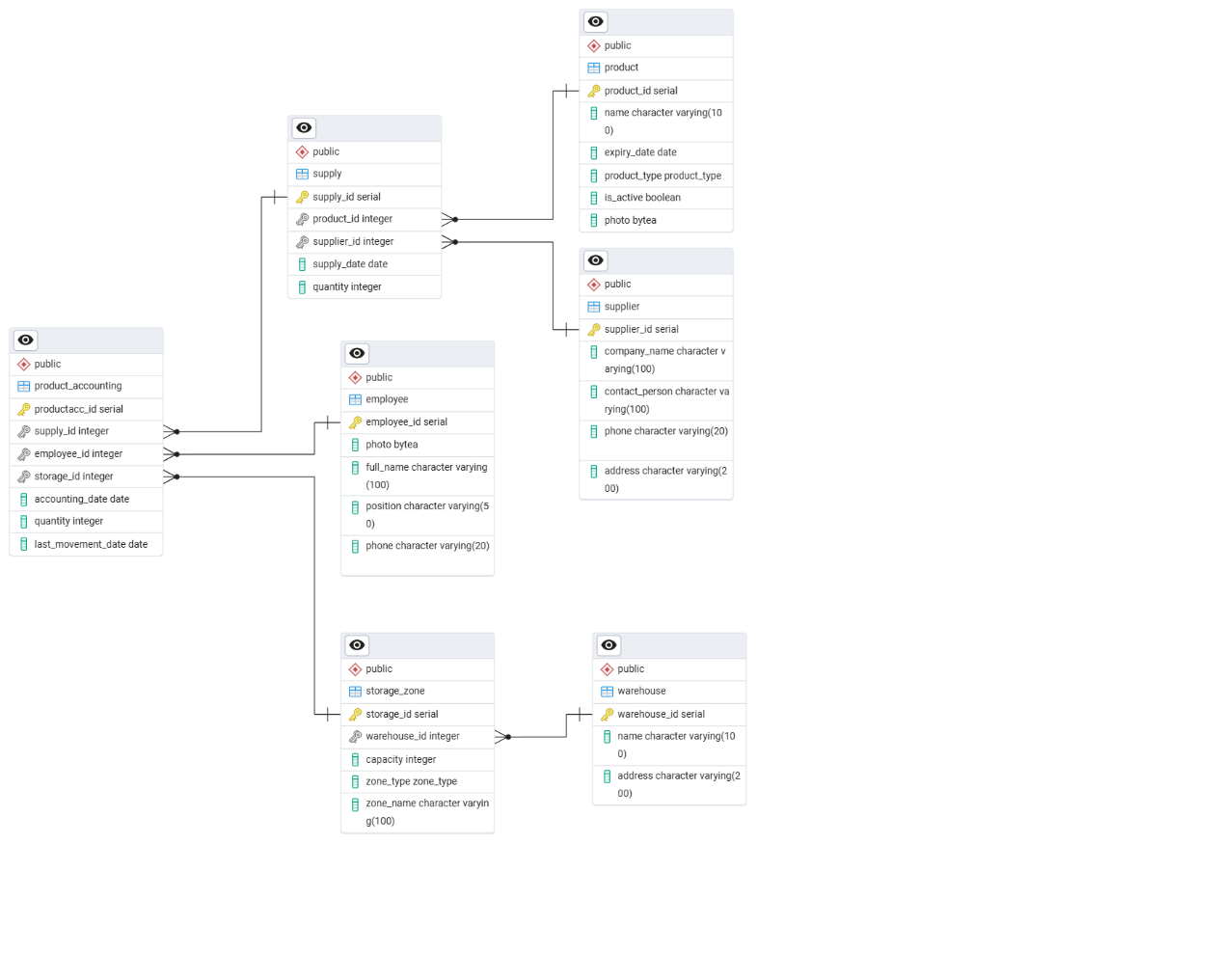


Рисунок 1.5 – ERD схема данных

Теперь мы можем писать запросы (рисунок 1.6-1.13) к БД, в которую заранее записали данные.

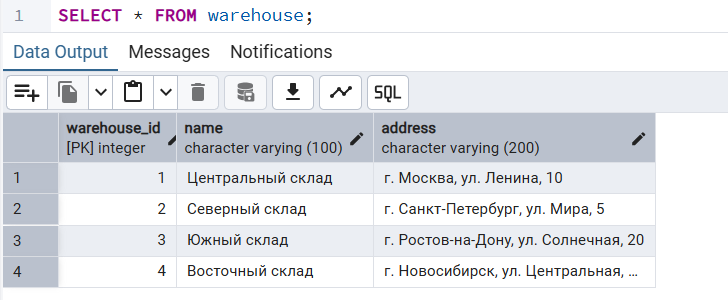


Рисунок 1.6 – Получение таблицы «Склад»

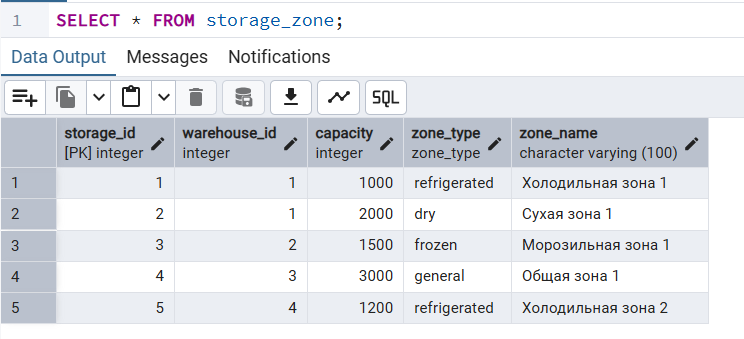


Рисунок 1.7 – Получение таблицы «Зона хранения»

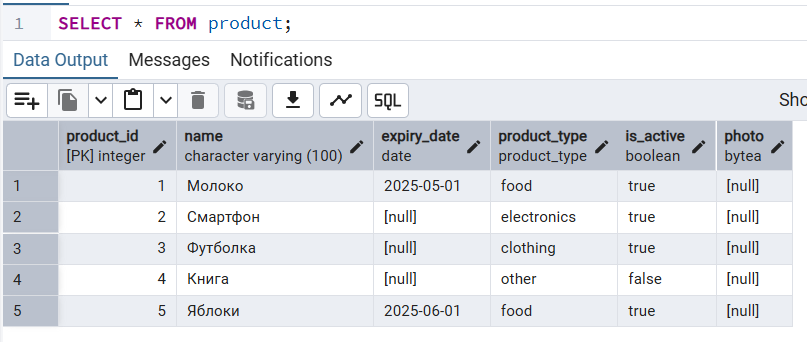


Рисунок 1.8 – Получение таблицы «Товар»

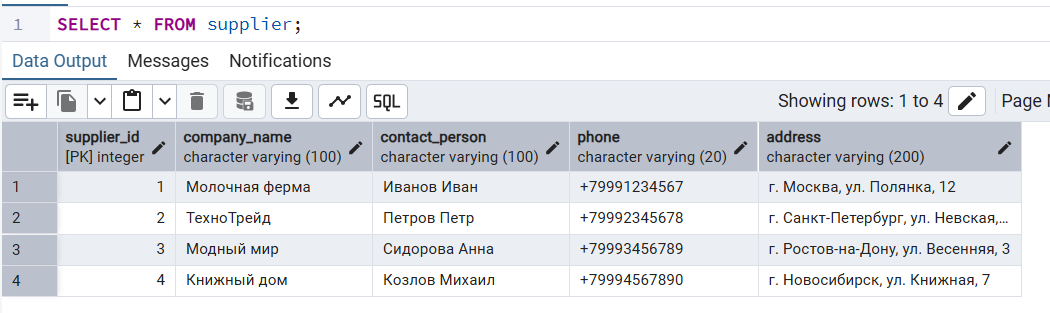


Рисунок 1.9 – Получение таблицы «Поставщик»

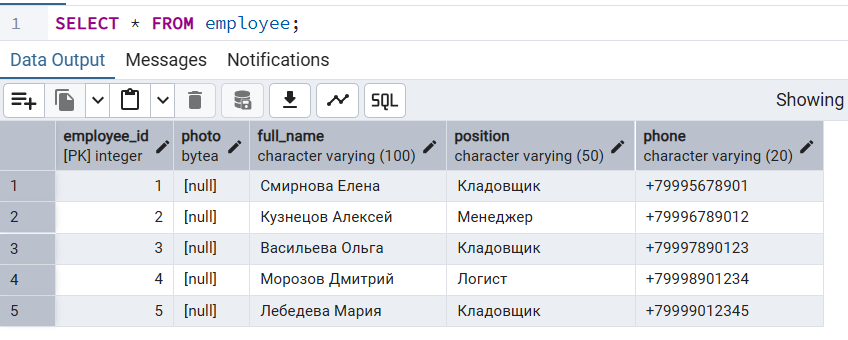


Рисунок 1.10 – Получение таблицы «Персонал»

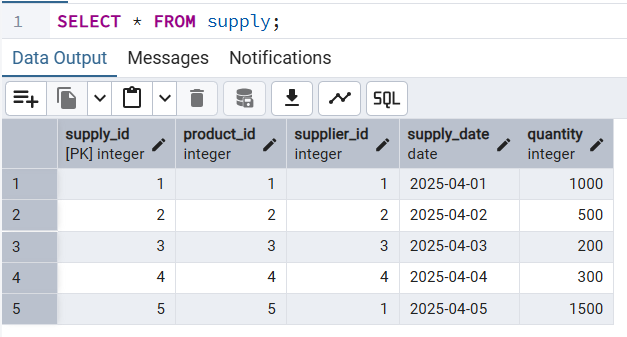


Рисунок 1.11 – Получение таблицы «Поставка»

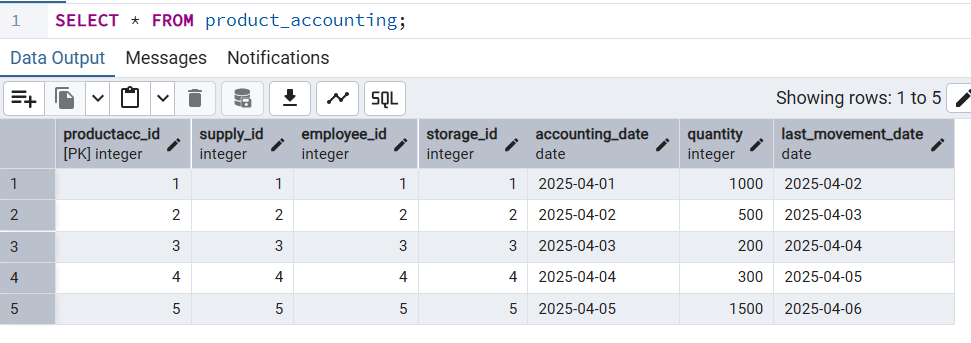


Рисунок 1.12 – Получение таблицы «Учет товара»

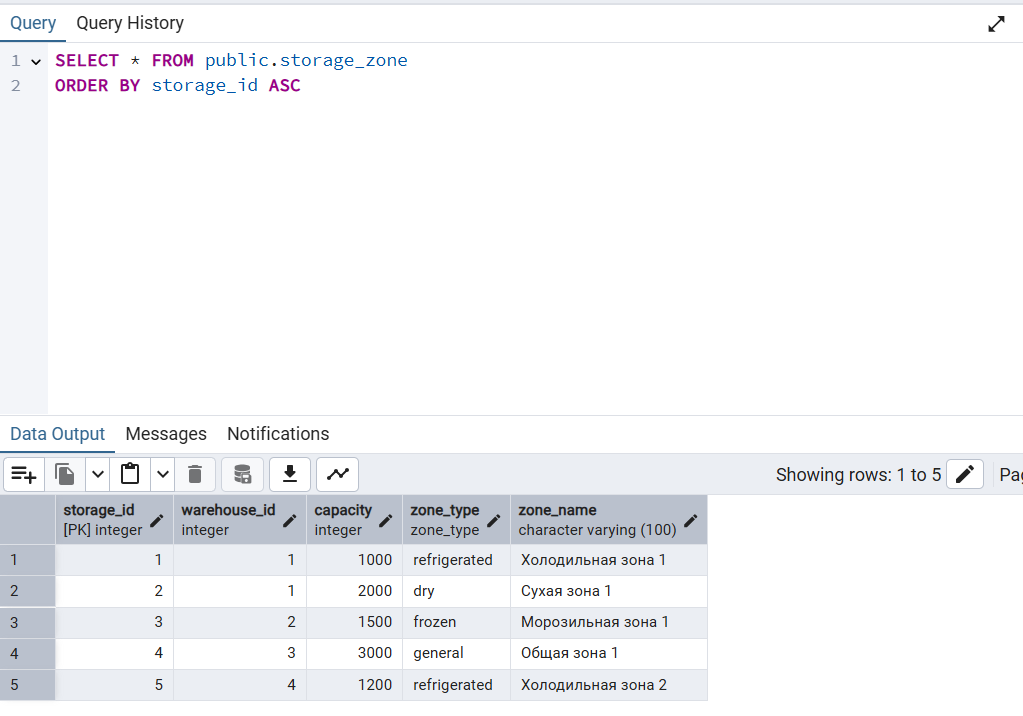


Рисунок 1.13 – Тестовый запрос к БД

БД проверяет на корректность ввод данных при добавлении элементов (рисунок 1.14), согласно выставленным ограничениям.

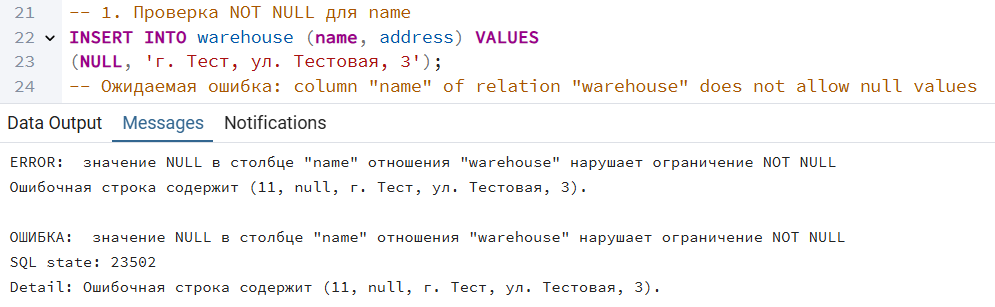


Рисунок 1.14 – Проверка добавления элементов

**Заключение**

В ходе выполнения лабораторной работы была спроектирована и реализована реляционная база данных, отражающая структуру школьной информационной системы. Были созданы основные таблицы, такие как employee, product, product\_accounting, storage\_zone, supplier, supply и warehouse, а также определены связи между ними с использованием внешних ключей.

Особое внимание было уделено применению ограничений целостности (PRIMARY KEY, FOREIGN KEY, NOT NULL), что обеспечило корректность и надёжность хранения данных. Для проверки функционирования базы данных были добавлены тестовые и некорректные данные, что позволило убедиться в правильности реализованных ограничений.

Таким образом, цель лабораторной работы была достигнута: приобретены теоретические знания и практические умения по проектированию, созданию и тестированию реляционных баз данных.