# **1. Постановка задачи**

Цель: ознакомление с MySQLServer

Оснащение: ПК, учебная и справочная литература.

Задание: Описание предметной области:

Торговая организация ООО «Торг» ведет торговлю различными товарами. Заказы клиентов составляются на основе заявок. В заказе клиента перечисляются товары с указанием количества. Каждый заказ имеет несколько статусов: Новый, В работе, Выполнен.

При продаже товаров фиксируется дата продажи, способ оплаты, сотрудник, клиент, перечисляются товары и их количество. При покупке покупателю можно предоставить возможность регистрации в электронную почту, которые впоследствии будут занесены в базу клиентов для предоставления скидки. Отчество является не обязательным для заполнения. В зависимости от спроса сотрудники организации могут менять цену товаров.

Любая единица товара обладает собственными артикулом, наименованием, категорией, ценой, изображением (не обязательное), единицей измерения, датой изготовления, сроком годности, фирмой и страной производителя.

Для сотрудников организации необходимо хранить глобальный уникальный идентификатор в формате GUID. Всякий раз, когда добавляется новый сотрудник оно должно генерироваться автоматически. Так же необходимо хранить ФИО, дата рождения, телефон, адрес (город, улица, дом, квартира). Отчество является не обязательным для заполнения.

На основе описания предметной области Вам необходимо спроектировать ER-диаграмму для информационной системы.

Обязательна 3 нормальная форма с обеспечением ссылочной целостности. При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи, определите ограничения внешних ключей, отражающие характер предметной области. ER - диаграмма должна быть представлена в формате .pdf и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Создайте базу данных, используя предпочтительную платформу, на сервере баз данных, который вам предоставлен. Создайте таблицы основных сущностей, атрибуты, отношения и необходимые ограничения. В любом случае созданные таблицы должны содержать начальные тестовые данные. В каждой таблице должно быть как минимум 2 записи, а в таблице заказы и покупки клиентов 10 записей.

Предусмотреть, чтобы один клиент совершил несколько покупок, а каждая покупка должна быть оформлена на несколько товаров.

Создайте процедуру, которая проверяет адрес электронной почты, хранящийся в базе данных, на корректность. Корректным является адрес, если он содержит допустимые символы в каждой из частей и части разделены сначала символом @ затем «.» (Формат адреса электронной почты: A-Z0-9 @ A-Z0-9 . A-Z0-9). Дополнительно необходимо проверить, чтобы в адресе отсутствовали символы, из-за которых часто возникают ошибки ( [ " < > ' ]). Процедура выводит все адреса электронных почт с указанием соответствующего признака валидности (1-валидный, 0 – не валидный).

Для хранения истории изменения цен на товары создайте таблицу HistoryCost со следующей структурой: Дата изменения, Товар, Старое значение цены, Новое значение цены). Создайте триггер, который при изменении цены будет записывать данные в таблицу HistoryCost.

Предполагается, что за один раз будет обновление только одной строки.

# **2. Практическая часть**

На рисунке 1 изображена ER-диаграмма базы данных на основе описания предметной области.

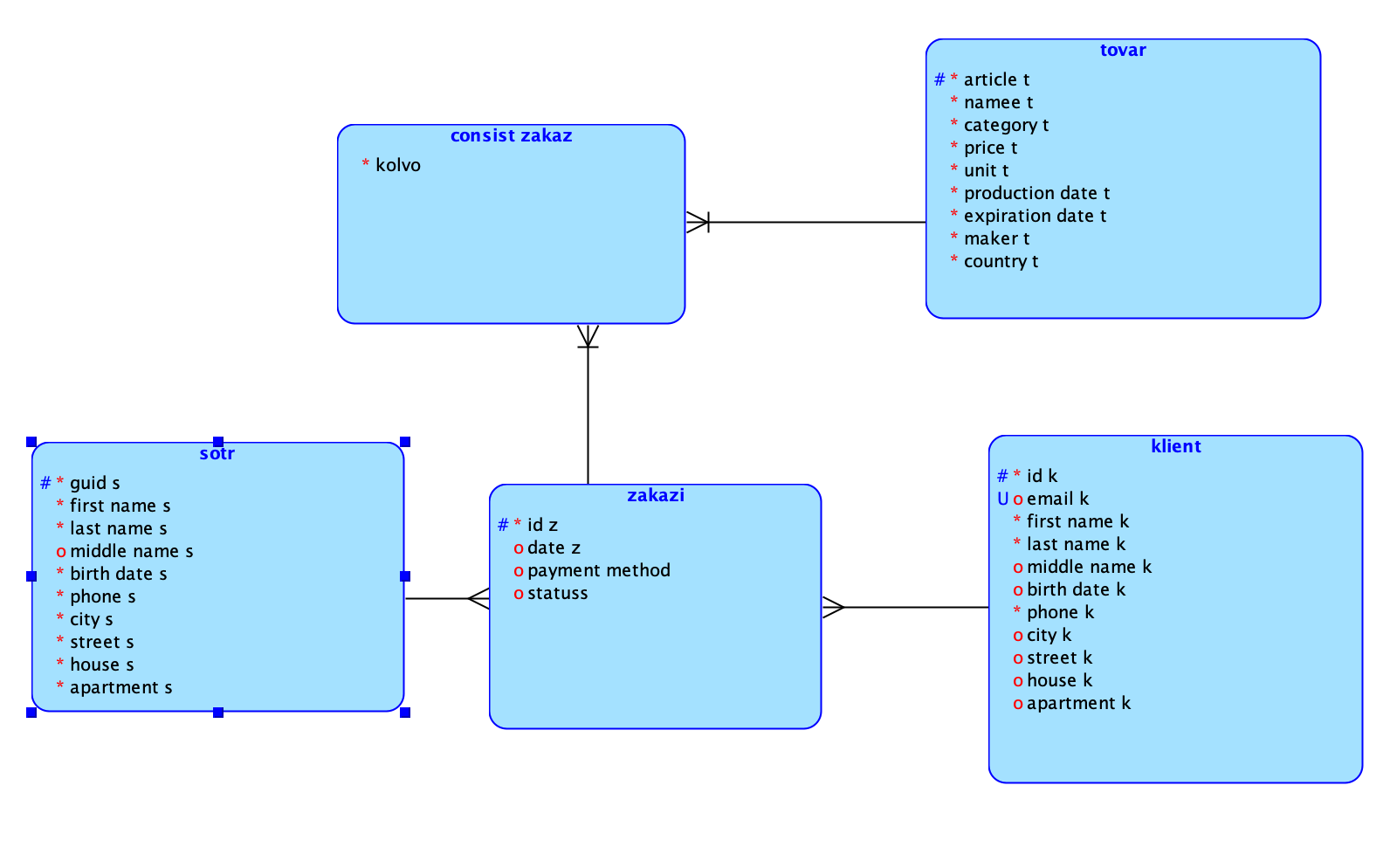


Рисунок 1 - ER-диаграмма базы данных

Для реализации физической базы данных в MySQL необходимо выполнить следующий SQL скрипт.

CREATE DATABASE TradeCompany;

USE TradeCompany;

CREATE TABLE klient (

id\_k INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,

email\_k VARCHAR(70) UNIQUE,

first\_name\_k VARCHAR(50) NOT NULL,

last\_name\_k VARCHAR(50) NOT NULL,

middle\_name\_k VARCHAR(60),

birth\_date\_k DATE,

phone\_k VARCHAR(16) NOT NULL,

city\_k VARCHAR(100),

street\_k VARCHAR(100),

house\_k VARCHAR(100),

apartment\_k VARCHAR(20)

);

CREATE TABLE sotr (

guid\_s CHAR(36) PRIMARY KEY default(uuid()),

first\_name\_s VARCHAR(50) NOT NULL,

last\_name\_s VARCHAR(50) NOT NULL,

middle\_name\_s VARCHAR(60),

birth\_date\_s DATE NOT NULL,

phone\_s VARCHAR(16) NOT NULL,

city\_s VARCHAR(100) NOT NULL,

street\_s VARCHAR(100) NOT NULL,

house\_s VARCHAR(100) NOT NULL,

apartment\_s VARCHAR(20) NOT NULL

);

CREATE TABLE zakazi (

id\_z INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,

id\_k INT NOT NULL,

guid\_s CHAR(36) NOT NULL,

date\_z DATE default(current\_date()),

payment\_method ENUM('Наличные', 'Безналичный'),

statuss ENUM('Новый', 'В работе', 'Выполнен'),

FOREIGN KEY (id\_k) REFERENCES klient(id\_k) ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY (guid\_s) REFERENCES sotr(guid\_s) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE tovar (

article\_t VARCHAR(255) PRIMARY KEY NOT NULL,

namee\_t VARCHAR(100) NOT NULL,

category\_t VARCHAR(100) NOT NULL,

price\_t DECIMAL(10, 2) NOT NULL,

unit\_t VARCHAR(50) NOT NULL,

production\_date\_t DATE NOT NULL,

expiration\_date\_t DATE NOT NULL,

maker\_t VARCHAR(100) NOT NULL,

country\_t VARCHAR(60) NOT NULL

);

CREATE TABLE consist\_zakaz (

id\_z INT NOT NULL,

article\_t VARCHAR(255) NOT NULL,

kolvo INT NOT NULL,

PRIMARY KEY (id\_z, article\_t),

FOREIGN KEY (id\_z) REFERENCES zakazi(id\_z) ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY (article\_t) REFERENCES tovar(article\_t) ON UPDATE CASCADE ON DELETE CASCADE

);

На рисунке 2 изображена физическая схема базы данных.

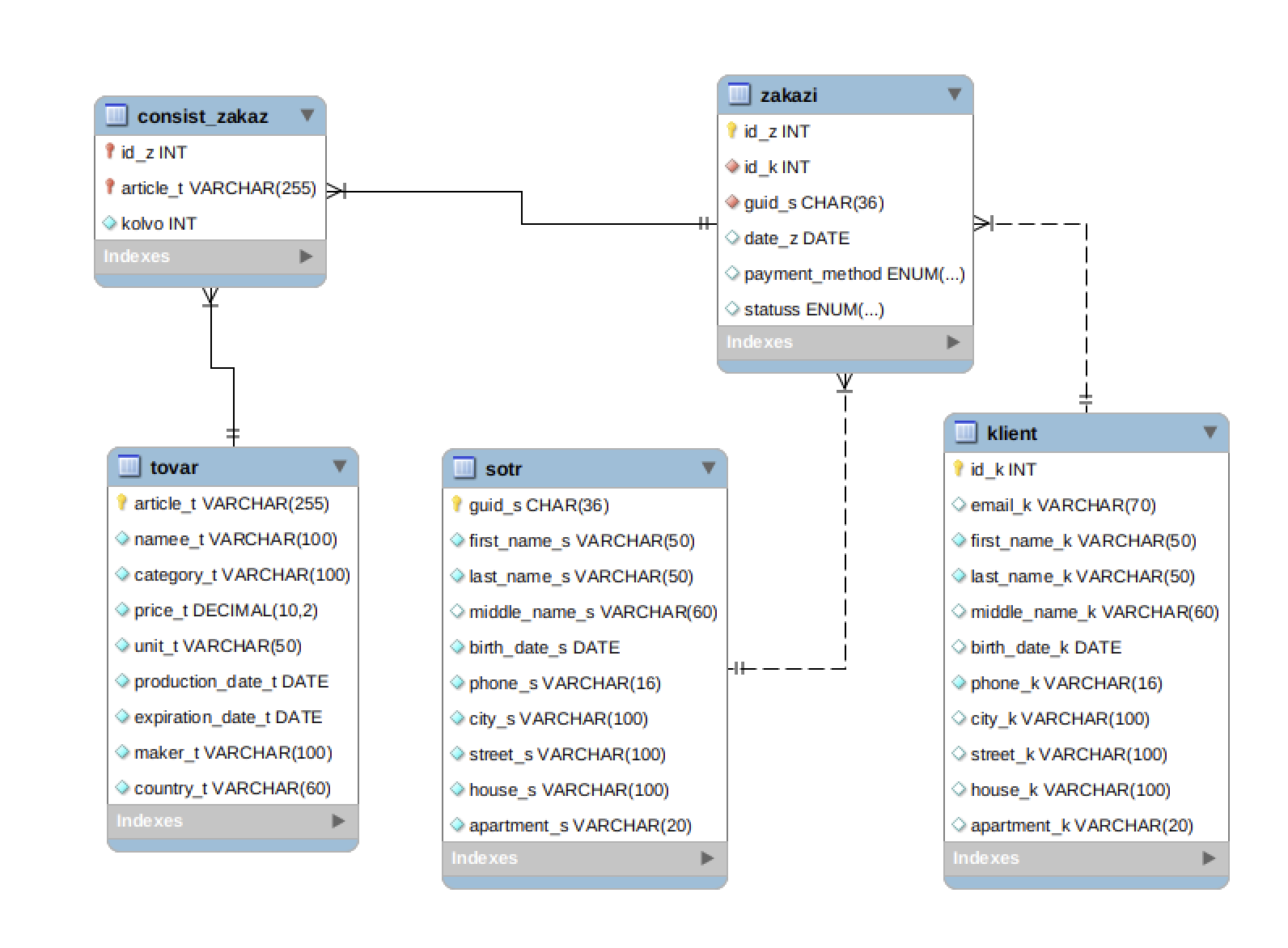


Рисунок 2 – Физическая схема базы данных

Для заполнения базы данных необходимо выполнить следующий SQL скрипт.

-- Заполнение таблицы klient

INSERT INTO klient (email\_k, first\_name\_k, last\_name\_k, middle\_name\_k, birth\_date\_k, phone\_k, city\_k, street\_k, house\_k, apartment\_k)

VALUES

('client1@example.com', 'Иван', 'Иванов', 'Иванович', '1990-01-01', '+79123456789', 'Москва', 'Ленина', '1', '1'),

('client2@example.com', 'Петр', 'Петров', 'Петрович', '1985-02-02', '+79123456788', 'Санкт-Петербург', 'Невский', '2', '2'),

('client3@example.com', 'Анна', 'Сидорова', 'Сидоровна', '1995-05-05', '+79123456781', 'Екатеринбург', 'Ленина', '10', '10'),

('client4example.com', 'Елена', 'Смирнова', 'Алексеевна', '1980-03-03', '+79123456787', 'Москва', 'Ленина', '3', '3');

-- Заполнение таблицы sotr

INSERT INTO sotr (first\_name\_s, last\_name\_s, middle\_name\_s, birth\_date\_s, phone\_s, city\_s, street\_s, house\_s, apartment\_s)

VALUES

('Сергей', 'Сергеев', 'Сергеевич', '1980-03-03', '+79123456787', 'Москва', 'Ленина', '3', '3'),

('Мария', 'Иванова', 'Ивановна', '1987-04-04', '+79123456786', 'Санкт-Петербург', 'Невский', '4', '4'),

('Елена', 'Сидорова', 'Сидоровна', '1990-06-06', '+79123456780', 'Екатеринбург', 'Ленина', '10', '10'),

('Антон', 'Петров', 'Петрович', '1985-07-07', '+79123456785', 'Санкт-Петербург', 'Пушкина', '5', '5');

-- Заполнение таблицы tovar

INSERT INTO tovar (article\_t, namee\_t, category\_t, price\_t, unit\_t, production\_date\_t, expiration\_date\_t, maker\_t, country\_t)

VALUES

('ART001', 'Товар 1', 'Категория 1', 100.00, 'шт', '2022-01-01', '2023-01-01', 'Производитель 1', 'Страна 1'),

('ART002', 'Товар 2', 'Категория 2', 200.00, 'шт', '2022-01-02', '2023-01-02', 'Производитель 2', 'Страна 2'),

('ART003', 'Товар 3', 'Категория 3', 300.00, 'шт', '2022-01-03', '2023-01-03', 'Производитель 3', 'Страна 3'),

('ART004', 'Товар 4', 'Категория 4', 400.00, 'шт', '2022-01-04', '2023-01-04', 'Производитель 4', 'Страна 4');

-- Заполнение таблицы zakazi

INSERT INTO zakazi (id\_k, guid\_s, date\_z, payment\_method, statuss)

VALUES

(1, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-01', 'Наличные', 'Новый'),

(2, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-02', 'Безналичный', 'В работе'),

(3, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-03', 'Наличные', 'Выполнен'),

(4, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-04', 'Безналичный', 'Новый'),

(1, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-05', 'Наличные', 'В работе'),

(2, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-06', 'Безналичный', 'Выполнен'),

(3, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-07', 'Наличные', 'Новый'),

(4, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-08', 'Безналичный', 'В работе'),

(1, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-09', 'Наличные', 'Выполнен'),

(2, (SELECT guid\_s FROM sotr ORDER BY RAND() LIMIT 1), '2022-01-10', 'Безналичный', 'Новый');

-- Заполнение таблицы consist\_zakaz

INSERT INTO consist\_zakaz (id\_z, article\_t, kolvo)

VALUES

(1, 'ART001', 2),

(1, 'ART002', 3),

(1, 'ART003', 1),

(2, 'ART004', 2),

(2, 'ART002', 3),

(3, 'ART002', 3),

(3, 'ART003', 1),

(3, 'ART004', 2),

(3, 'ART001', 2),

(4, 'ART002', 3),

(4, 'ART003', 1),

(4, 'ART004', 2);

На рисунках 3 – 7 изображены заполненные таблицы базы данных.

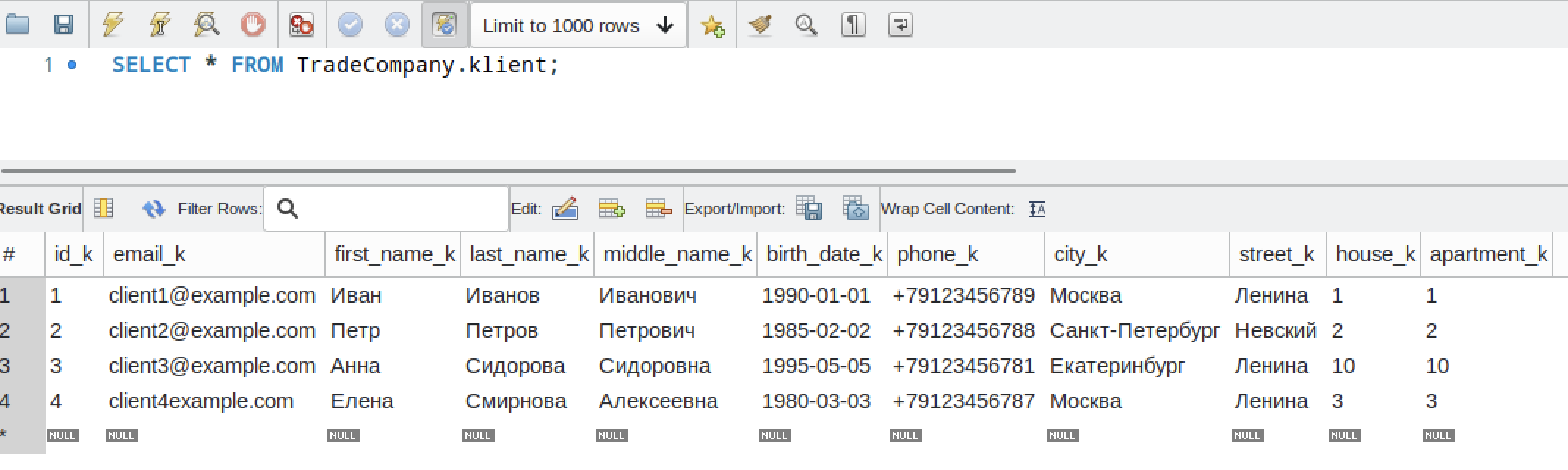


Рисунок 3 – Заполненная таблица клиент

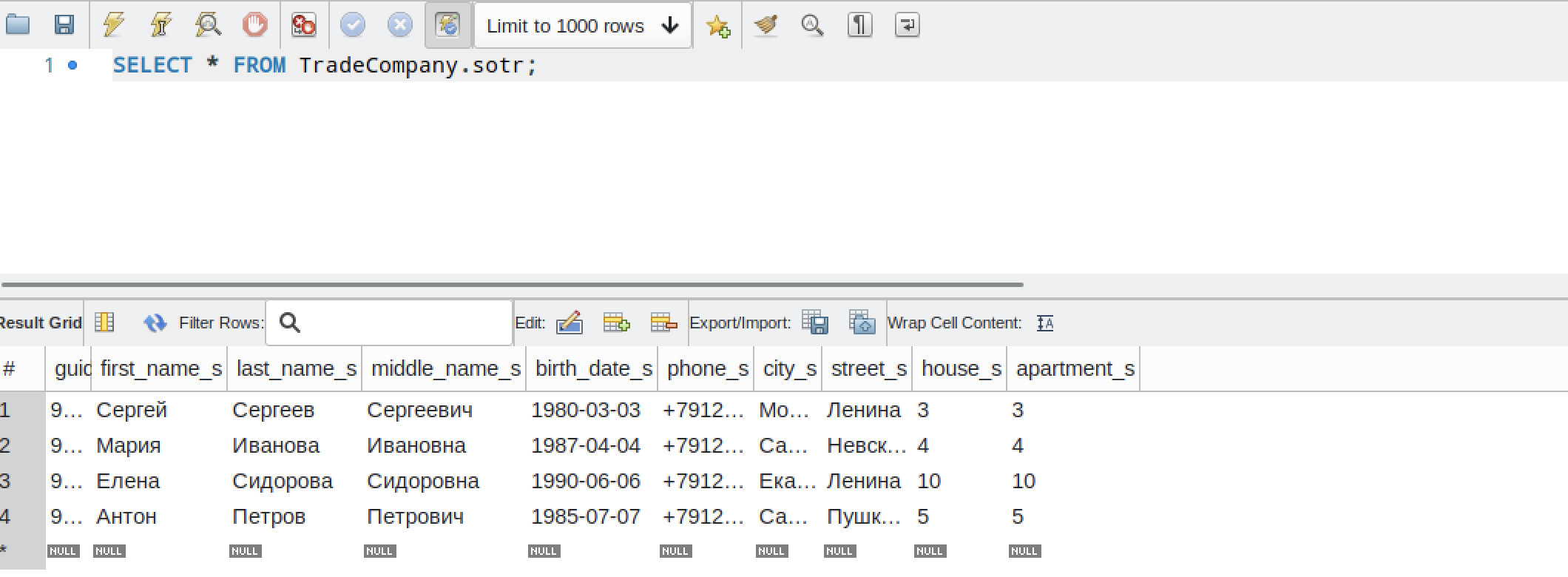


Рисунок 4 – Заполненная таблица сотрудник

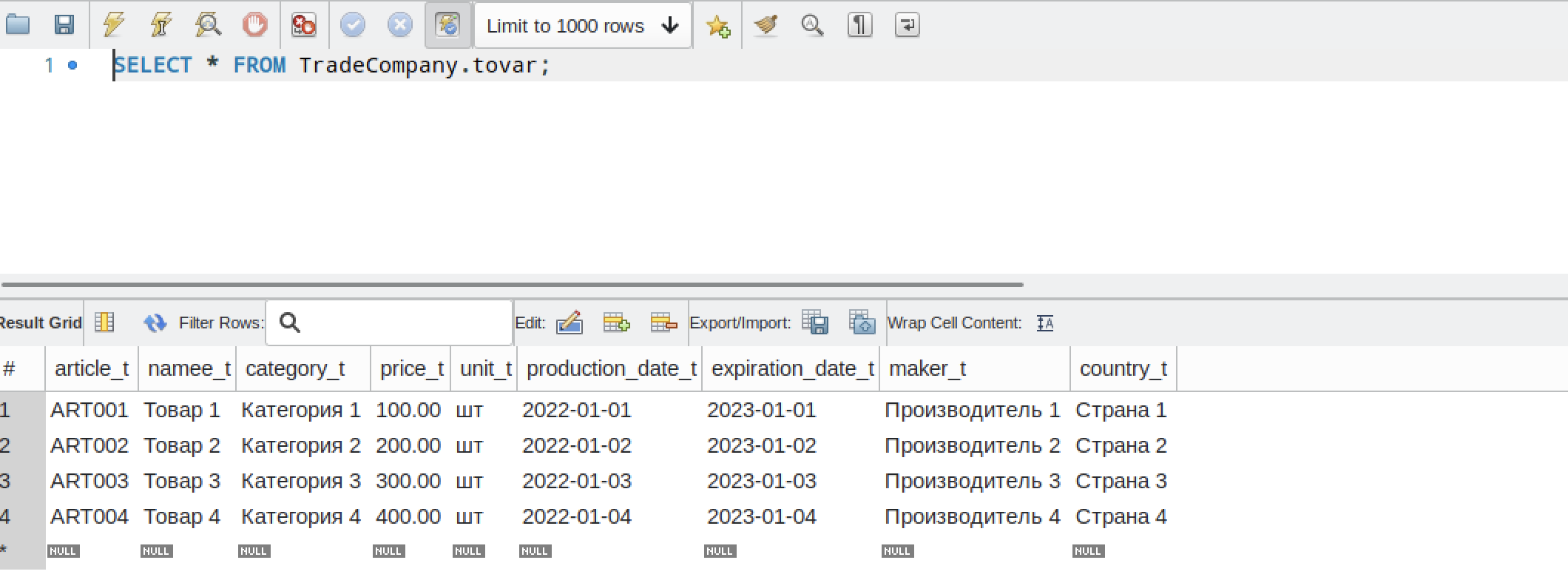


Рисунок 5 – Заполненная таблица товар

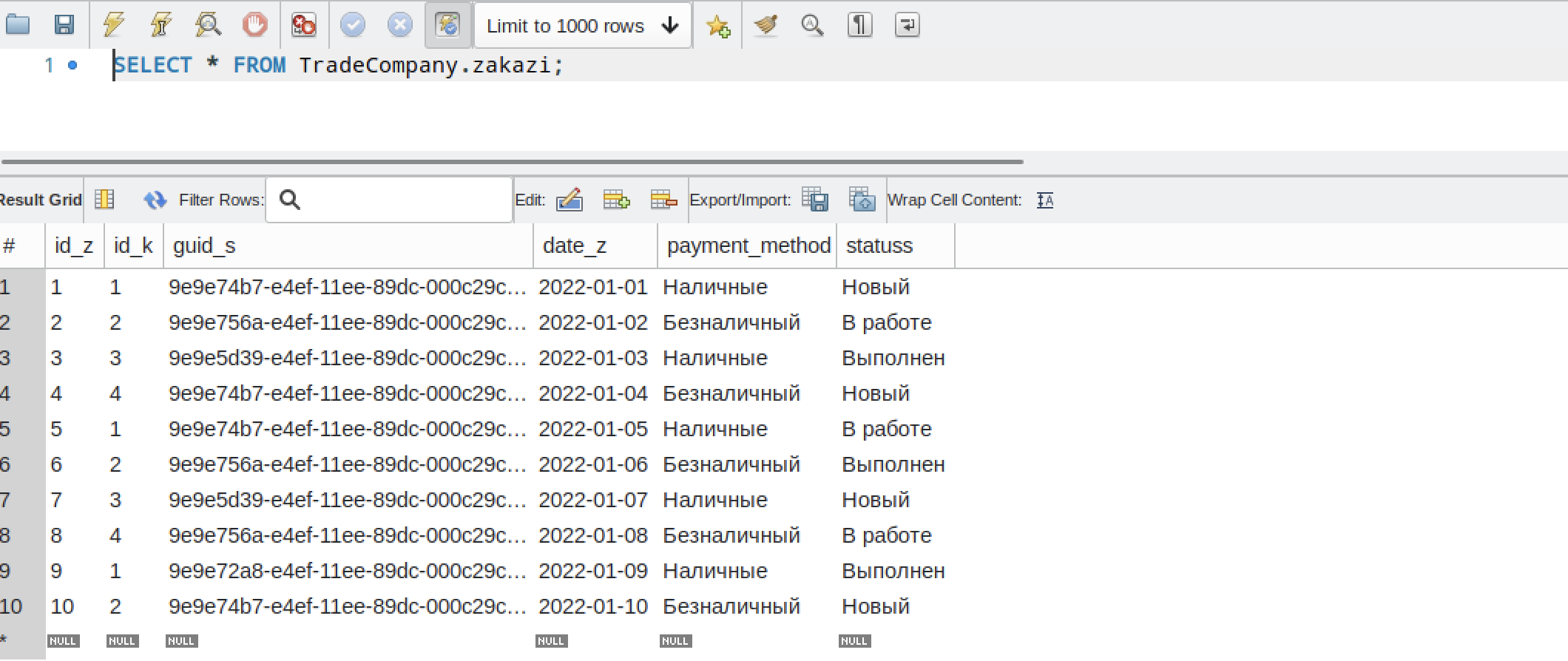


Рисунок 6 – Заполненная таблица заказы

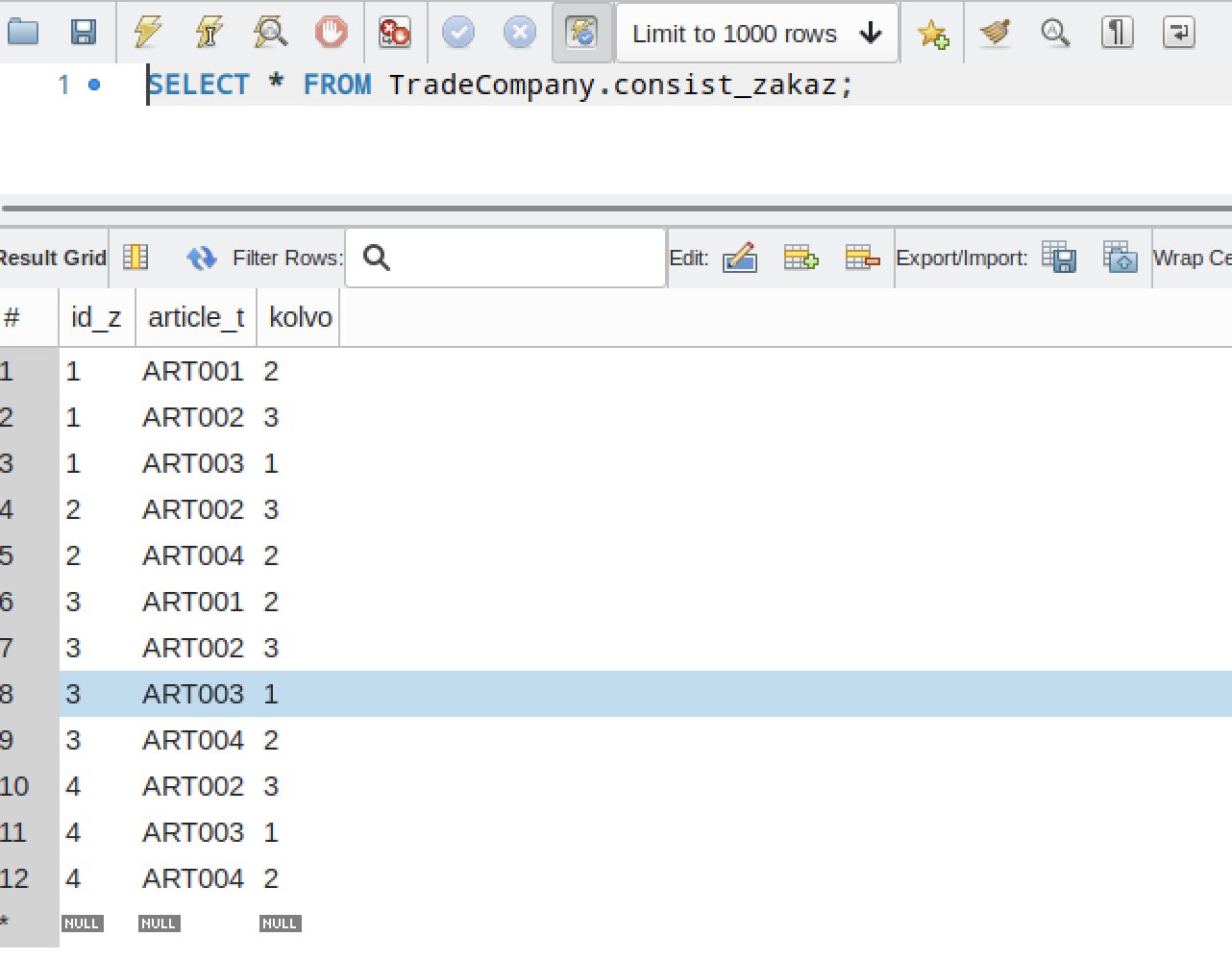


Рисунок 7 – Заполненная таблица состав заказа

Создать процедуру, которая проверяет адрес электронной почты, хранящийся в базе данных, на корректность. Корректным является адрес, если он содержит допустимые символы в каждой из частей и части разделены сначала символом @ затем «.» (Формат адреса электронной почты: A-Z0-9 @ A-Z0-9 . A-Z0-9). Дополнительно необходимо проверить, чтобы в адресе отсутствовали символы, из-за которых часто возникают ошибки ( [ " < > ' ]). Процедура выводит все адреса электронных почт с указанием соответствующего признака валидности (1-валидный, 0 – не валидный) можно с помощью следующего SQL скрипта.

DELIMITER //

CREATE PROCEDURE CheckEmail()

BEGIN

SELECT email\_k AS Email,

CASE

WHEN email\_k REGEXP '^[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$' THEN 1

ELSE 0

END AS IsValid

FROM klient;

END //

DELIMITER ;

Вызвать процедуру можно с помощью следующего SQL скрипта.

CALL CheckEmail();

На рисунке 8 изображен результат выполнения SQL скрипта.

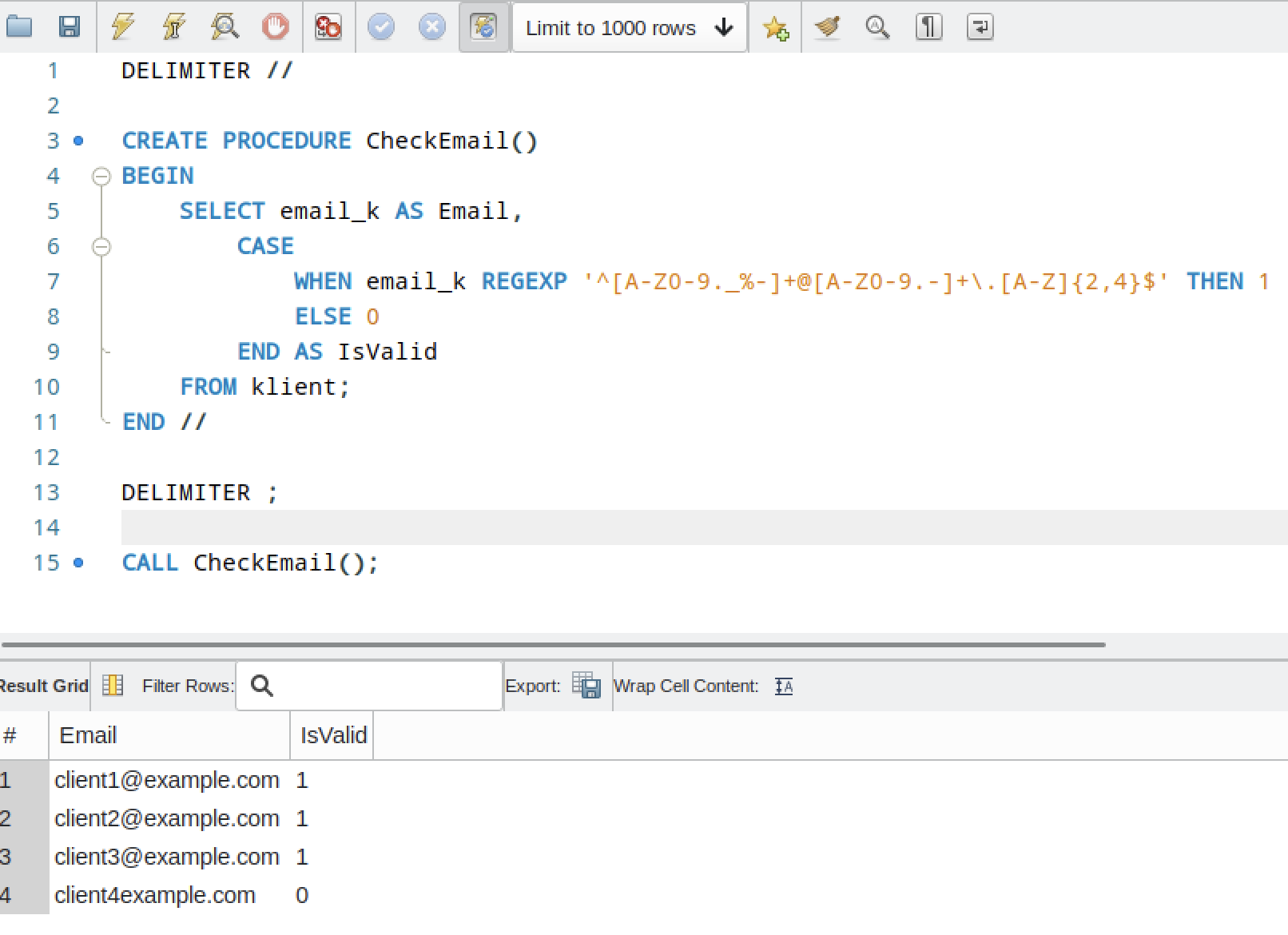


Рисунок 8 – Результат выполнения SQL скрипта

Для хранения истории изменения цен на товары создать таблицу HistoryCost со следующей структурой: Дата изменения, Товар, Старое значение цены, Новое значение цены) можно с помощью следующего sql скрипта.

CREATE TABLE HistoryCost (

id INT AUTO\_INCREMENT PRIMARY KEY,

change\_date DATE NOT NULL default(curdate()),

article\_t VARCHAR(255) NOT NULL,

old\_price DECIMAL(10, 2) NOT NULL,

new\_price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (article\_t) REFERENCES tovar(article\_t)

);

На рисунке 9 изображена физическая структура таблицы история цены.

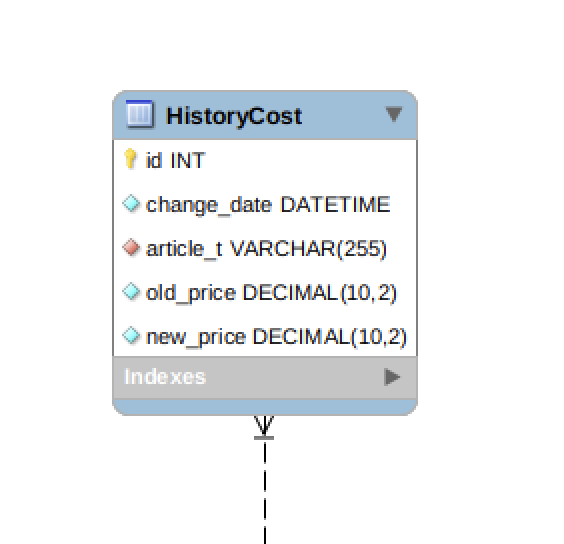


Рисунок 9 – Физическая структура таблицы история цены

Создать триггер, который при изменении цены будет записывать данные в таблицу HistoryCost можно с помощью следующего sql скрипта.

DELIMITER //

CREATE TRIGGER tovar\_update

BEFORE UPDATE ON tovar

FOR EACH ROW

BEGIN

IF NEW.price\_t <> OLD.price\_t THEN

INSERT INTO HistoryCost (change\_date, article\_t, old\_price, new\_price)

VALUES (NOW(), OLD.article\_t, OLD.price\_t, NEW.price\_t);

END IF;

END;

//

DELIMITER ;

Обновить цену товара и посмотреть результат выполнения триггера можно с помощью следующего sql скрипта.

UPDATE `TradeCompany`.`tovar` SET `price\_t` = '80.00' WHERE (`article\_t` = 'ART001');

SELECT \* FROM TradeCompany.HistoryCost;

На рисунке 10 изображен результат выполнения SQL скрипта.

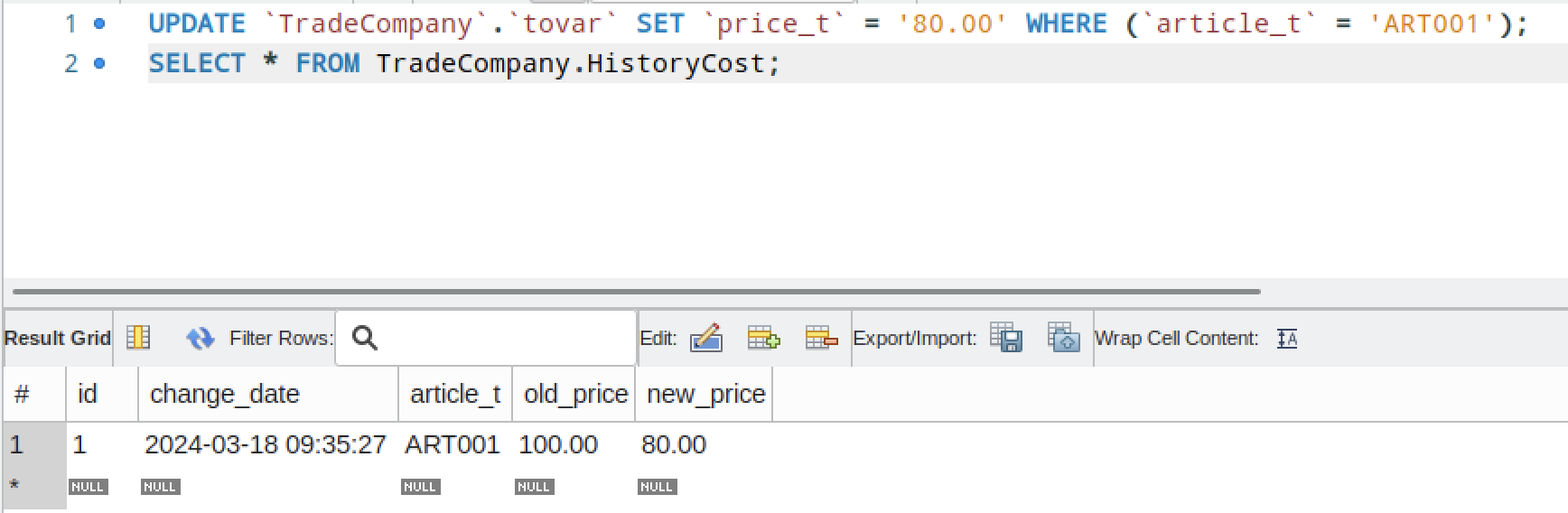


Рисунок 10 – Результат выполнения SQL скрипта

**ВЫВОД**

В ходе выполнения практической работы были получены практические навыки по работе в MySQL.