Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Направление (специальность) Информатика и вычислительная техника  
Кафедра САПР и ПК

Дисциплина \_\_\_\_\_\_\_Базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | Утверждаю  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | «\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г. |

**ЗАДАНИЕ**

**на курсовую работу (проект)**

Студент\_\_\_\_Мантуленко Андрей Игоревич\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Группа\_\_\_ИВТ-263\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема: Проектирование и разработка базы данных интернет-магазина электроники и приложения для взаимодействия с ней

Утверждена приказом от «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. № \_\_\_\_\_\_\_\_\_

2. Срок представления работы (проекта) к защите «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

3. Содержание расчетно-пояснительной записки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Перечень графического материала: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

Руководитель работы (проекта)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Соколов А.А.\_\_\_

подпись, дата инициалы и фамилия

Задание принял к исполнению Мантуленко А.И.

подпись, дата инициалы и фамилия

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра САПР и ПК

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе (проекту)**

по дисциплине \_\_\_\_\_\_\_Базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему Проектирование и разработка базы данных интернет-магазина электроники и приложения для взаимодействия с ней

Студент\_\_\_ Мантуленко А.И.

(фамилия, имя, отчество)

Группа\_\_\_ИВТ-263\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы (проекта) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соколов А.А.\_

(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

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

(подпись и дата подписания) (инициалы и фамилия)

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

(подпись и дата подписания) (инициалы и фамилия)

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

(подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата подписания) (инициалы и фамилия)

Волгоград 2021 г.

Содержание

[Концептуальное проектирование базы данных 4](#_Toc74570066)

[1. Описание предметной области 4](#_Toc74570067)

[2. ER-диаграмма 8](#_Toc74570068)

[Логическое проектирование БД 9](#_Toc74570069)

[1. Реляционная модель 9](#_Toc74570070)

[2. Набор данных 9](#_Toc74570071)

[Физическое проектирование БД 12](#_Toc74570072)

[1. Реляционная схема в MySQL 12](#_Toc74570073)

[2. Реляционная схема (из второй лабораторной работы) 13](#_Toc74570074)

[3. Таблицы 13](#_Toc74570075)

[Реализация SQL-запросов базы данных 16](#_Toc74570076)

[Индексы, хранимые процедуры, представления 25](#_Toc74570077)

[Взаимодействие с БД из приложения 37](#_Toc74570078)

[Ссылка на репозиторий: 41](#_Toc74570079)

[Список литературы 41](#_Toc74570080)

# Концептуальное проектирование базы данных

# Описание предметной области

Выбранная предметная область – интернет-магазин электроники. Доступ к нему будут иметь клиенты и сотрудники магазина. У клиентов есть возможность просмотра товаров, а также оформления заказа. У сотрудников же помимо просмотра есть возможность редактирования товаров и заказов. Сотрудниками являются директор, администраторы и менеджеры.

Для проектирования базы данных выберем точку зрения администратора.

1. Функциональные и нефункциональные требования

Функциональные требования определяют, какого рода операции разрабатываемая программа может выполнять по запросу пользователя, внешней системы или автоматически: транзакционные, оперативные, аналитические, плановые.

Транзакционные (задачи учета):

* Добавить новый товар;
* Изменить информацию о товаре;
* Добавить товар в корзину (заказ);
* Оформить заказ;
* Изменить информацию о заказе;
* Удалить товар;
* Удалить заказ.

Справочные (оперативные):

* Показать невыполненные заказы;
* Показать заказы с доставкой;
* Показать заказы с самовывозом;
* Показать отзывы о товаре;
* Показать все товары выбранной категории.

Справочные расчетные (аналитические):

* Показать статистику продаж за месяц (количество продаж за месяц);
* Показать статистику продаж за год (количество продаж за год);
* Показать общий рейтинг товара по отзывам;
* Показать наличие конкретного товара во всех магазинах;
* Показать количество заказов конкретного клиента.

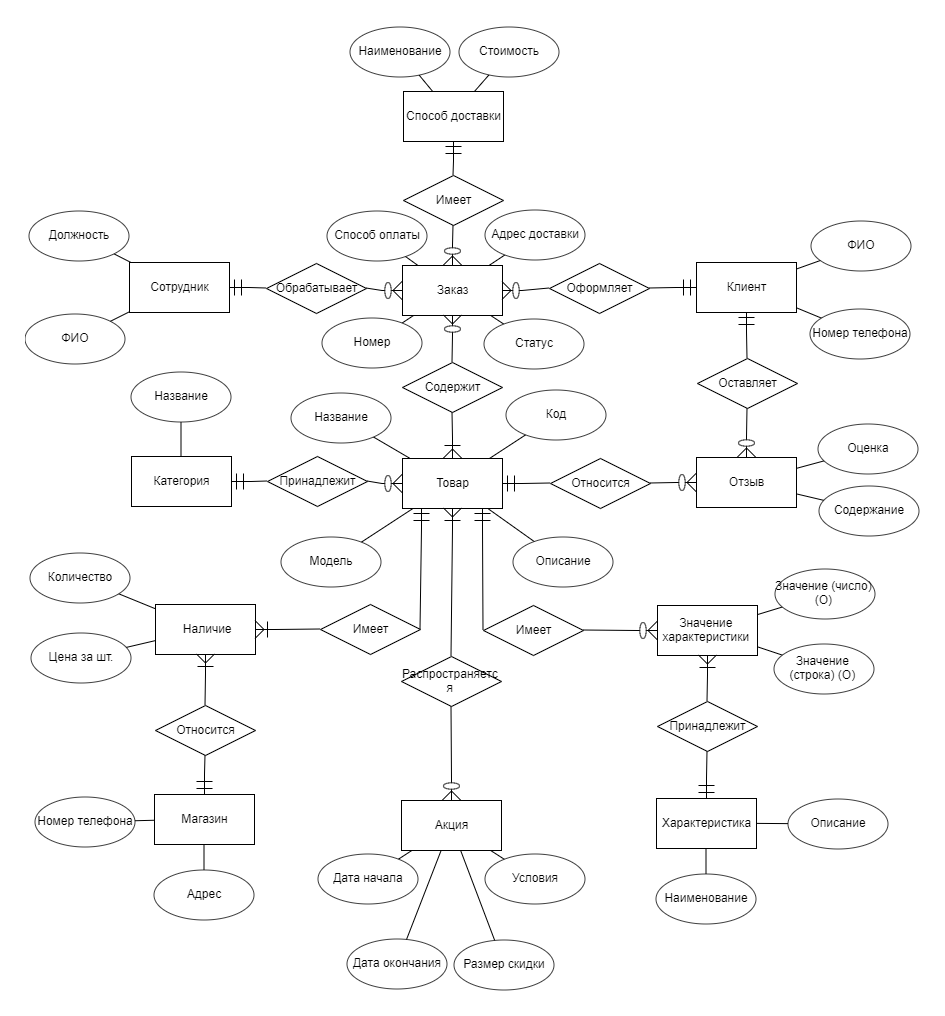
Типы сущностей:

* Сотрудник
* ФИО (строка)
* Должность (строка)
* Заказ
* Номер (целое число)
* Статус (строка)
* Способ получения (строка)
* Способ оплаты (строка)
* Адрес доставки/самовывоза
* Клиент
* ФИО (строка)
* Номер телефона (строка)
* Товар
* Название (строка)
* Модель (строка)
* Код (целое число)
* Категория
* Название (строка)
* Отзыв
* Оценка (целое число)
* Содержание (строка)
* Характеристика
* Наименование (строка)
* Описание (строка)
* Значение характеристики
* Значение (целое число)
* Значение (строка)
* Наличие
* Количество (целое число)
* Цена за шт. (целое число)
* Магазин
* Адрес (строка)
* Номер телефона (строка)
* Акция
* Дата начала (строка)
* Дата окончания (строка)
* Размер скидки (вещественное число)
* Условия (строка)
* Способ доставки
* Наименование
* Стоимость

Список связей:

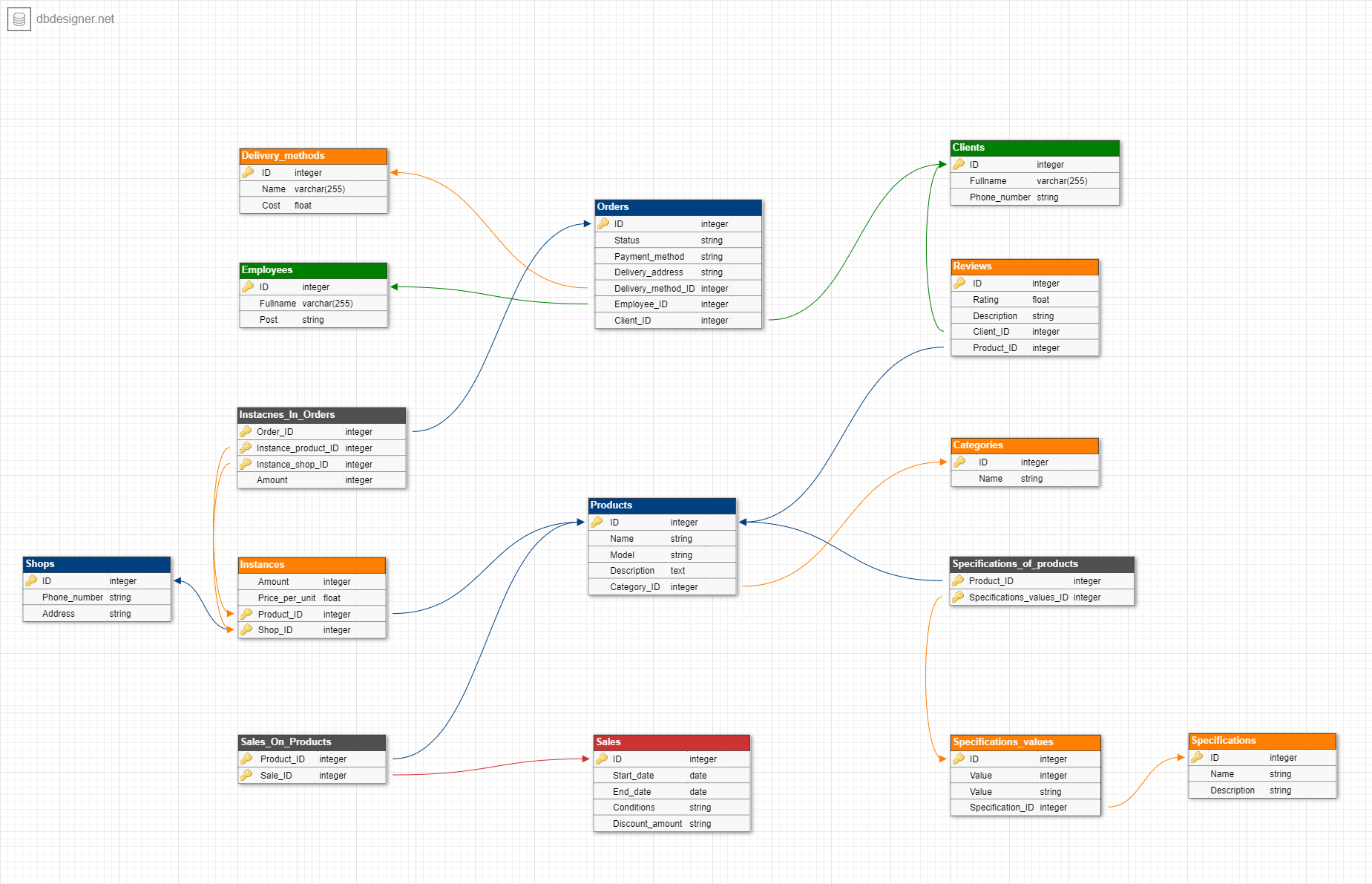
* **Сотрудник** *обрабатывает* **заказ**, который *оформлен* **клиентом**;
* **Заказ** *содержит* различные **товары**;
* **Клиент** *оставляет* **отзыв**, *относящийся* к **товару**;
* **Товар** *принадлежит* **категории**;
* **Товар** *имеет* **наличие**;
* **Наличие** *относится* к **магазину**;
* На **товар** *распространяются* **акции**;
* **Товар** *имеет* **значения характеристик**, *принадлежащих* **характеристикам**.

# ER-диаграмма



# Логическое проектирование БД

# Реляционная модель



# Набор данных

Categories:



Clients:



Delivery\_methods:



Emploees:



Instances:



Instances\_in\_orders:



Orders:



Products:



Reviews:



Sales:



Sales\_on\_products:



Shops:



Specifications:



Specifications\_of\_products:

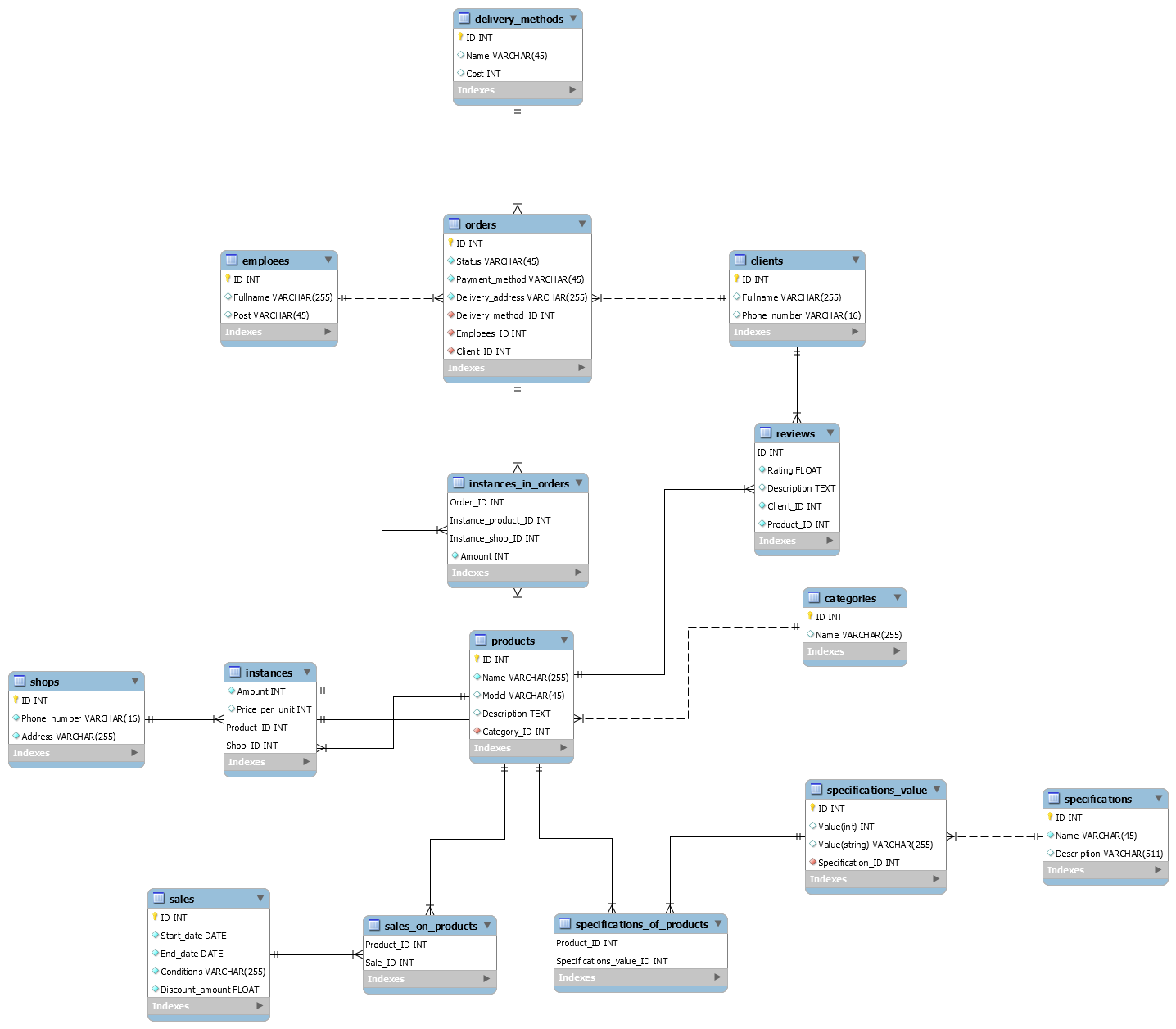


Specifications\_values:

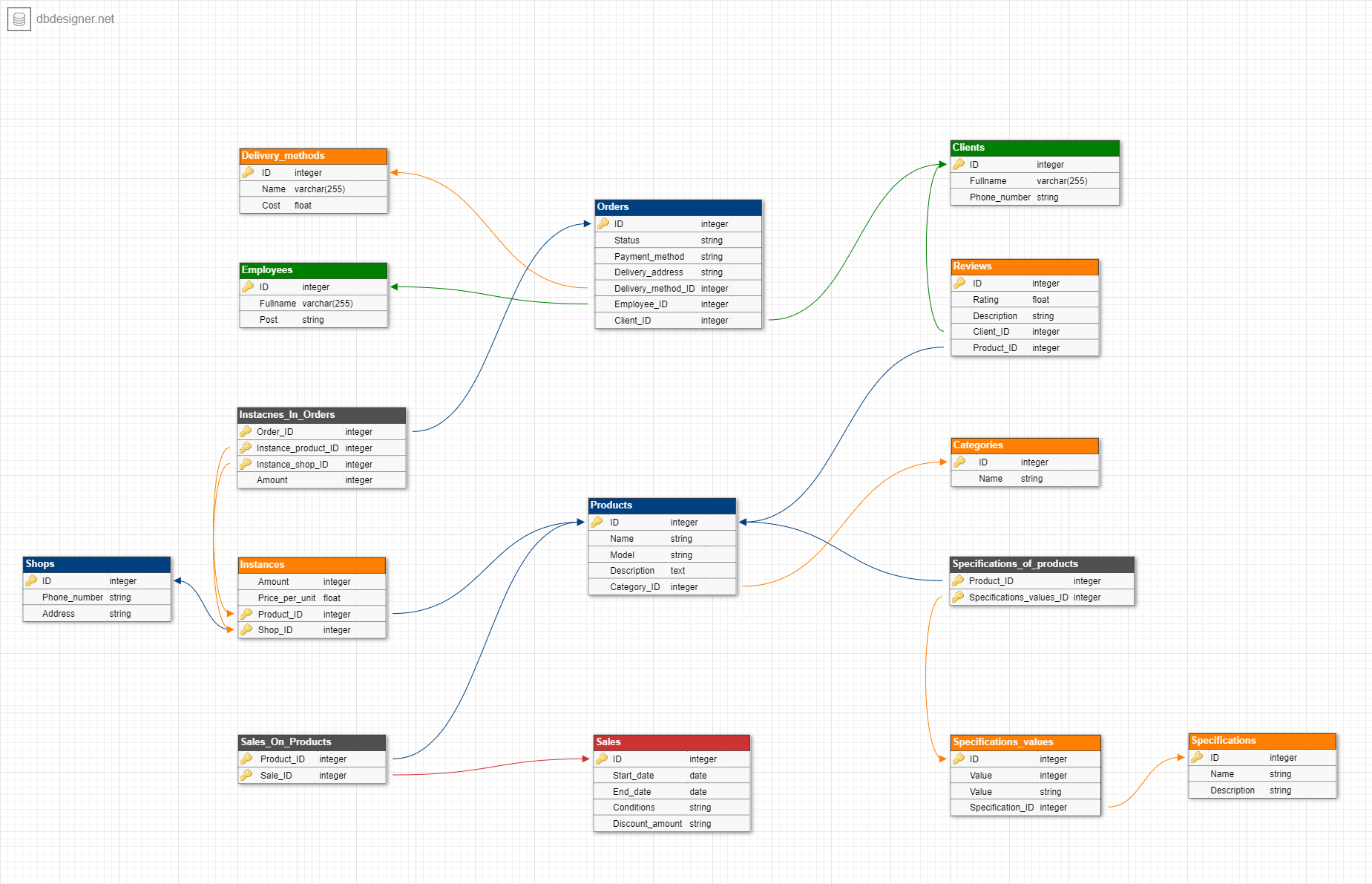


# Физическое проектирование БД

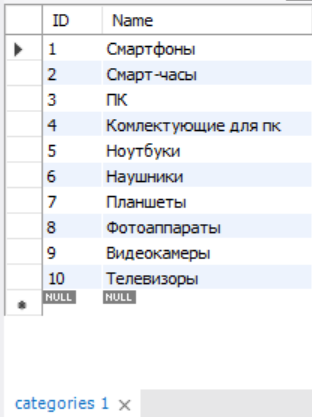
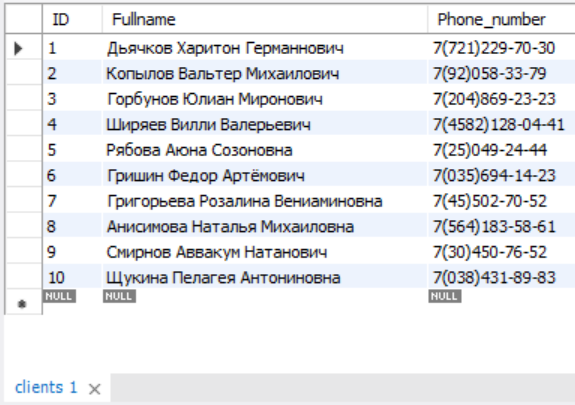
# Реляционная схема в MySQL

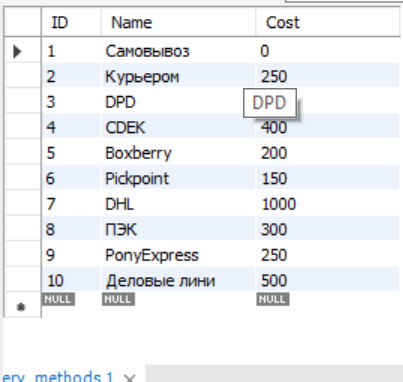
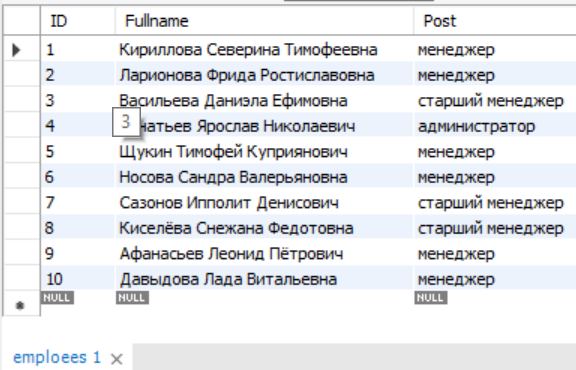


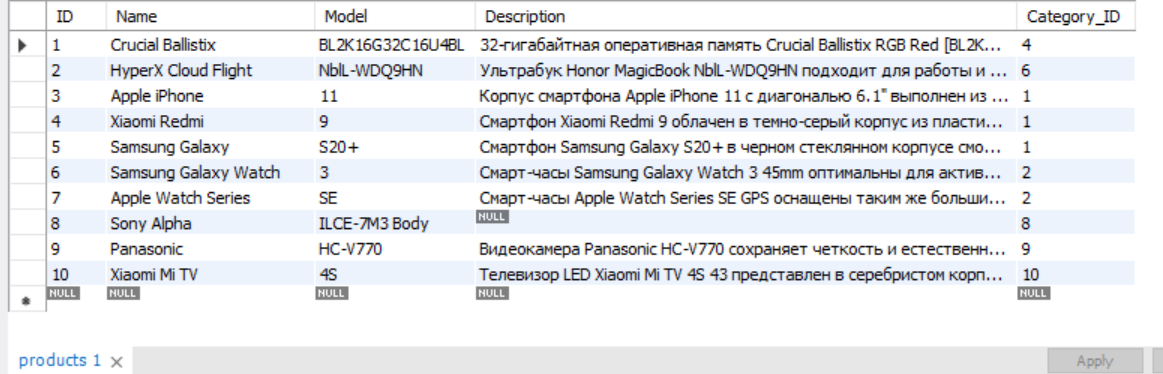
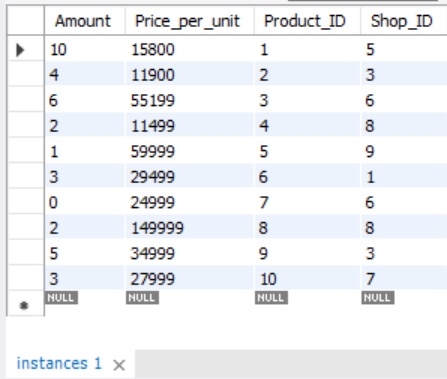
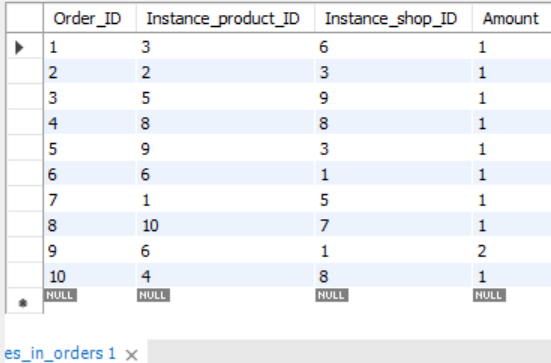
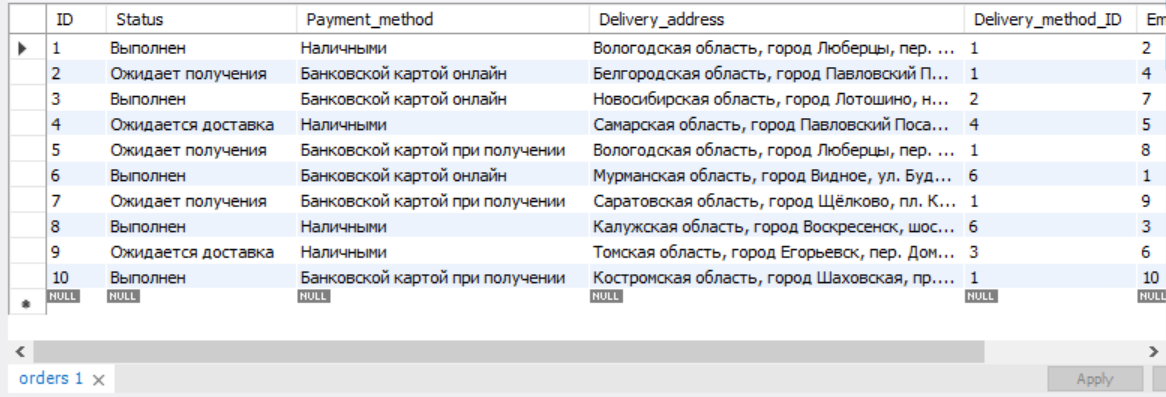
# Реляционная схема (из второй лабораторной работы)

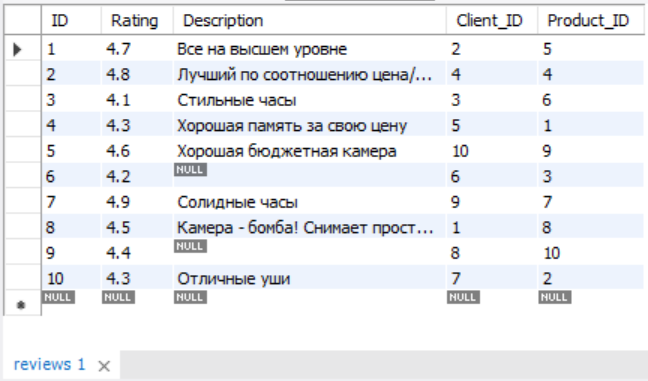
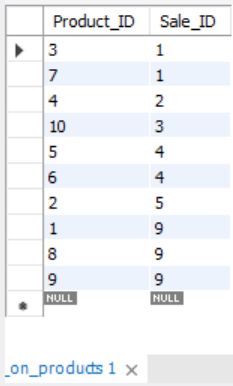


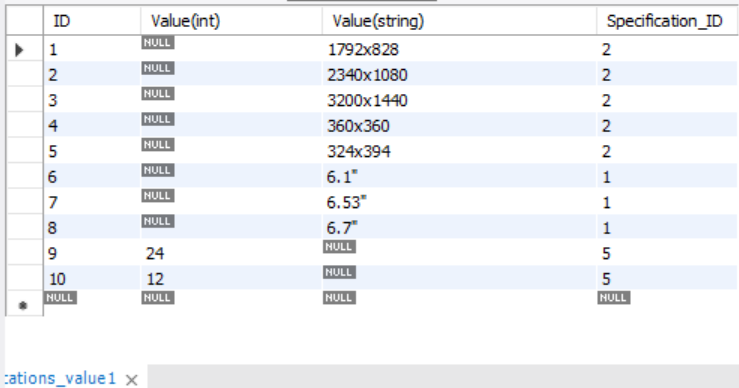
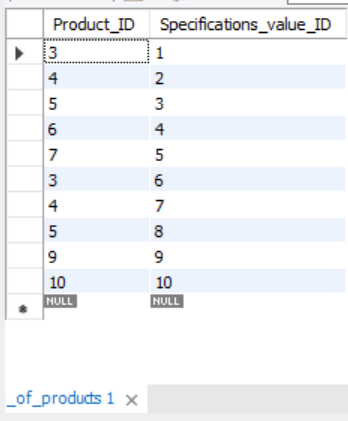
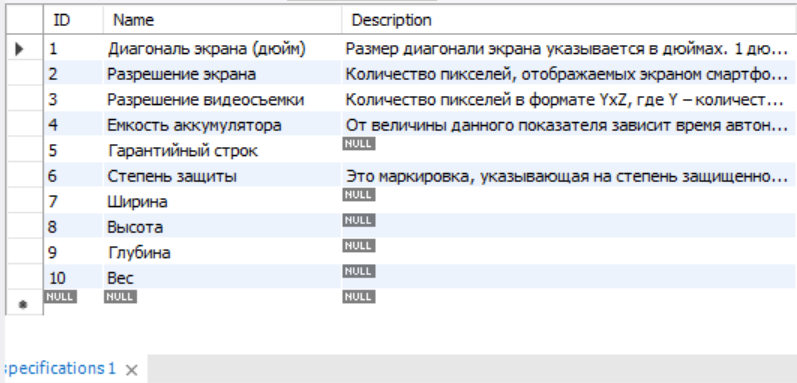
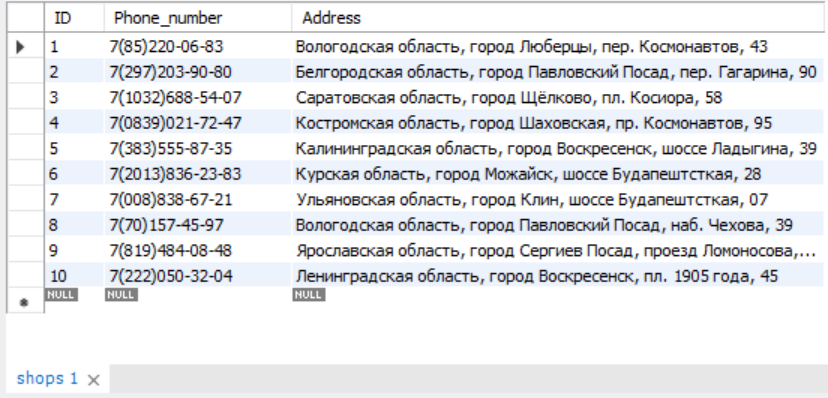
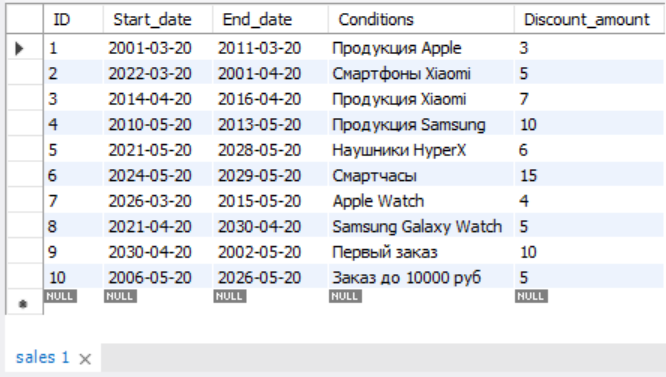
# 3. Таблицы











# Реализация SQL-запросов базы данных

UPDATE:

1. UPDATE clients SET Phone\_number = '7(835)942-48-03' WHERE ID = 6;
2. UPDATE products SET `Description` = 'Описание товара' WHERE ID = 8;
3. UPDATE instances SET Price\_per\_unit = 10900 WHERE Product\_ID = 2 AND Shop\_ID = 3;
4. UPDATE sales SET Discount\_amount = 5 WHERE ID = 1;
5. UPDATE orders SET Payment\_method = 'Банковской картой онлайн' WHERE ID = 2;

DELETE:

1. DELETE FROM instances\_in\_orders WHERE Order\_ID = 1;
2. DELETE FROM orders WHERE ID = 1;
3. DELETE FROM reviews WHERE ID = 2;
4. DELETE FROM clients WHERE ID = 4;
5. DELETE FROM sales WHERE ID = 7;

WHERE:

1. SELECT `Name`, Model FROM products WHERE Category\_ID = 1;
2. SELECT ID FROM orders WHERE `Status` != 'Выполнен';
3. SELECT ID FROM orders WHERE Delivery\_method\_ID != 1;
4. SELECT ID FROM orders WHERE Delivery\_method\_ID = 1;
5. SELECT \* FROM reviews WHERE Product\_ID = 8;
6. SELECT \* FROM instances\_in\_orders WHERE Instance\_product\_ID = 6;
7. SELECT \* FROM instances WHERE Product\_ID = 4;
8. SELECT \* FROM orders WHERE `Status` = 'Ожидается доставка';
9. SELECT COUNT(\*) FROM sales WHERE End\_date BETWEEN '2028-01-01' AND '2030-12-20';
10. SELECT COUNT(\*) FROM orders WHERE Client\_ID = 3;
11. SELECT COUNT(\*) FROM orders WHERE Emploees\_ID = 6;
12. SELECT \* FROM specifications WHERE `Description` IS NULL;
13. SELECT \* FROM reviews WHERE `Description` IS NOT NULL;
14. SELECT \* FROM specifications\_of\_products WHERE Product\_ID = 4;
15. SELECT \* FROM instances WHERE Shop\_ID = 8;
16. SELECT \* FROM specifications\_value WHERE Specification\_ID = 2;
17. SELECT COUNT(\*) FROM specifications\_value WHERE Specification\_ID = 1;
18. SELECT Product\_ID FROM instances WHERE Shop\_ID = 8;
19. SELECT \* FROM reviews WHERE Product\_ID = 4 AND Rating > 4;
20. SELECT \* FROM reviews WHERE Client\_ID = 5;
21. SELECT COUNT(\*) FROM reviews WHERE Client\_ID = 8;
22. SELECT \* FROM emploees WHERE Post = 'менеджер';

LIKE:

1. SELECT \* FROM products WHERE `Name` LIKE '%Apple%';
2. SELECT \* FROM shops WHERE Address LIKE 'Вологодская область%';
3. SELECT \* FROM emploees WHERE Post LIKE '%менеджер%';
4. SELECT \* FROM orders WHERE Delivery\_address LIKE 'Вологодская область%';
5. SELECT \* FROM products WHERE `Name` LIKE '%Smasung%';
6. SELECT \* FROM orders WHERE Delivery\_date LIKE '2021-04%';
7. SELECT COUNT(\*) FROM shops WHERE Address LIKE 'Курская область%';
8. SELECT \* FROM sales WHERE Conditions LIKE '%Watch%';
9. SELECT \* FROM specifications WHERE `Name` LIKE '%Разрешение%';
10. SELECT \* FROM products WHERE `Name` LIKE '%Xiaomi%';

INSERT:

1. INSERT INTO Test(column1, column2) SELECT `Name`, Model FROM products WHERE `Name` LIKE '%Xiaomi%';
2. INSERT INTO Test(column3) SELECT ID FROM orders WHERE `Status` != 'Выполнен';
3. INSERT INTO Test(column1) SELECT `Name` FROM categories WHERE ID = 2;

JOIN:

1. SELECT specifications\_value.`Value(int)`, specifications\_value.`Value(string)`, specifications.`Name`, specifications.`Description` FROM specifications\_value

JOIN specifications ON specifications\_value.Specification\_ID = specifications.ID;

1. SELECT `Name`, Model, specifications\_value.`Value(int)`, specifications\_value.`Value(string)` FROM products

RIGHT JOIN specifications\_of\_products

ON specifications\_of\_products.Product\_ID = products.ID

RIGHT JOIN specifications\_value

ON specifications\_value.ID = Specifications\_value\_ID;

1. SELECT clients.Fullname, orders.ID, orders.`Status`, orders.Delivery\_date FROM clients

JOIN orders ON Client\_ID = clients.ID;

1. SELECT orders.ID, orders.`Status`, orders.Delivery\_date, delivery\_methods.Name, delivery\_methods.Cost FROM orders

JOIN delivery\_methods ON Delivery\_method\_ID = delivery\_methods.ID;

1. SELECT orders.ID, orders.`Status`, emploees.Fullname FROM orders

JOIN emploees ON Emploees\_ID = emploees.ID;

1. SELECT products.ID, products.`Name`, products.Model, categories.`Name` FROM products

JOIN categories ON Category\_ID = categories.ID;

1. SELECT products.ID, products.`Name`, products.Model, instances.Amount FROM products

JOIN instances ON products.ID = instances.Product\_ID;

1. SELECT products.`Name`, products.Model, reviews.Rating, reviews.`Description` FROM products

JOIN reviews ON products.ID = reviews.Product\_ID

JOIN clients ON reviews.Client\_ID = clients.ID;

1. SELECT products.ID, products.`Name`, products.Model, instances.Amount, shops.Address FROM products

LEFT JOIN instances ON products.ID = instances.Product\_ID

LEFT JOIN shops ON instances.Shop\_ID = shops.ID;

1. SELECT products.Name, products.Model, sales.Conditions, sales.Discount\_amount FROM products

NATURAL JOIN sales\_on\_products

NATURAL JOIN sales;

1. SELECT \* FROM instances

JOIN instances\_in\_orders

ON instances.Product\_ID = instances\_in\_orders.Instance\_product\_ID AND

instances.Shop\_ID = instances\_in\_orders.Instance\_shop\_ID

JOIN orders ON instances\_in\_orders.Order\_ID = orders.ID;

1. SELECT \* FROM sales CROSS JOIN products;
2. SELECT instances.Amount, instances.Price\_per\_unit, instances.Product\_ID, shops.Address, shops.Phone\_number FROM instances

JOIN shops ON instances.Shop\_ID = shops.ID;

1. SELECT specifications\_value.`Value(int)`, specifications\_value.`Value(string)`, specifications.`Name`, specifications.`Description` FROM specifications\_value

LEFT JOIN specifications ON specifications\_value.Specification\_ID = specifications.ID;

1. SELECT `Name`, Model, specifications\_of\_products.Specifications\_value\_ID FROM products

RIGHT JOIN specifications\_of\_products

ON specifications\_of\_products.Product\_ID = products.ID;

1. SELECT `Name`, Model, specifications\_of\_products.Product\_ID FROM products
2. RIGHT JOIN specifications\_of\_products

ON specifications\_of\_products.Product\_ID = products.ID;

1. SELECT specifications\_value.Specification\_ID, specifications\_value.`Value(int)`, specifications\_value.`Value(string)` FROM specifications

LEFT JOIN specifications\_value

ON specifications\_value.Specification\_ID = specifications.ID;

1. SELECT clients.Fullname, orders.ID, orders.`Status`, orders.Delivery\_date FROM clients

LEFT OUTER JOIN orders ON Client\_ID = clients.ID;

1. SELECT products.ID, products.`Name`, products.Model, categories.`Name` FROM products

RIGHT OUTER JOIN categories ON Category\_ID = categories.ID;

1. SELECT \* FROM products FULL JOIN reviews;

GROUP BY:

1. SELECT \* FROM products GROUP BY products.Category\_ID;
2. SELECT AVG(Rating) FROM reviews GROUP BY reviews.Product\_ID;
3. SELECT \* FROM reviews GROUP BY reviews.Client\_ID;
4. SELECT COUNT(\*), orders.`Status` FROM orders GROUP BY orders.`Status`;
5. SELECT COUNT(\*), orders.Payment\_method FROM orders GROUP BY orders.Payment\_method;
6. SELECT COUNT(\*), orders.Emploees\_ID FROM orders GROUP BY orders.Emploees\_ID;
7. SELECT COUNT(\*), orders.Client\_ID FROM orders GROUP BY orders.Client\_ID;
8. SELECT COUNT(\*), orders.Delivery\_address FROM orders GROUP BY orders.Delivery\_address;
9. SELECT MIN(instances.Price\_per\_unit), instances.Product\_ID, instances.Shop\_ID FROM instances GROUP BY instances.Product\_ID;
10. SELECT COUNT(\*) FROM instances\_in\_orders GROUP BY instances\_in\_orders.Instance\_shop\_ID;
11. SELECT COUNT(\*), specifications\_of\_products.Product\_ID FROM specifications\_of\_products GROUP BY specifications\_of\_products.Product\_ID;
12. SELECT COUNT(\*), orders.ID FROM orders GROUP BY orders.Delivery\_date;
13. SELECT COUNT(\*), specifications\_value.Specification\_ID FROM specifications\_value GROUP BY specifications\_value.Specification\_ID;
14. SELECT \* FROM instances GROUP BY instances.Product\_ID HAVING instances.Shop\_ID = 8;
15. SELECT \* FROM orders GROUP BY orders.`Status` HAVING orders.Delivery\_method\_ID = 1;
16. SELECT COUNT(\*), sales.Discount\_amount FROM sales GROUP BY sales.Discount\_amount;
17. SELECT products.`Name`, products.Model FROM products GROUP BY products.`Name`;
18. SELECT SUM(Price\_per\_unit \* Amount), instances.Shop\_ID FROM instances GROUP BY instances.Product\_ID;
19. SELECT MAX(instances.Price\_per\_unit), instances.Product\_ID, instances.Shop\_ID FROM instances GROUP BY instances.Product\_ID;
20. SELECT MAX(Rating), Product\_ID FROM reviews GROUP BY reviews.Product\_ID;

UNION:

1. SELECT orders.Delivery\_address FROM orders UNION ALL SELECT shops.Address FROM shops;
2. SELECT clients.Fullname FROM clients UNION SELECT emploees.Fullname FROM emploees;
3. SELECT sales.Conditions FROM sales UNION SELECT products.Name;

ALL, ANY, EXISTS:

1. SELECT ALL \* FROM products WHERE products.Category\_ID = 1;
2. SELECT ALL \* FROM orders WHERE orders.`Status` NOT LIKE '%Выполнен%';
3. SELECT ALL \* FROM emploees WHERE emploees.Post LIKE '%менеджер%';
4. SELECT \* FROM instances WHERE EXISTS (SELECT Price\_per\_unit FROM instances WHERE Price\_per\_unit < 20000);
5. SELECT \* FROM orders WHERE EXISTS (SELECT orders.`Status` FROM orders WHERE orders.`Status` LIKE '%Выполнен%');

GROUP\_CONCAT:

1. SELECT GROUP\_CONCAT(products.Category\_ID) FROM products;
2. SELECT GROUP\_CONCAT(orders.`Status`) FROM orders;

Сложные запросы:

-- Общая стоимость всех товаров проданных конкретным сотрудником

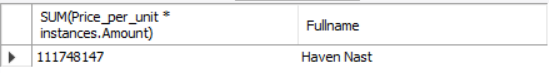
1. SELECT SUM(Price\_per\_unit \* instances.Amount), emploees.Fullname FROM instances\_in\_orders

JOIN instances ON Instance\_product\_ID = instances.Product\_ID AND Instance\_shop\_ID = instances.Shop\_ID

JOIN orders ON Order\_ID = orders.ID

JOIN emploees ON Emploees\_ID = emploees.ID

WHERE emploees.Fullname = 'Haven Nast';



-- Общая количество денег, потраченных конкретным клиентом

1. SELECT SUM(Price\_per\_unit \* instances.Amount), clients.Fullname FROM instances\_in\_orders

JOIN instances ON Instance\_product\_ID = instances.Product\_ID AND Instance\_shop\_ID = instances.Shop\_ID

JOIN orders ON Order\_ID = orders.ID

JOIN clients ON Client\_ID = clients.ID

WHERE clients.Fullname = 'Laural Fawke';



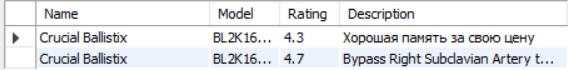
-- Все отзывы о конкретном товаре

1. SELECT products.`Name`, products.Model, reviews.Rating, reviews.`Description` FROM products

JOIN reviews ON products.ID = reviews.Product\_ID

JOIN clients ON reviews.Client\_ID = clients.ID

WHERE products.`Name` = "Crucial Ballistix";



-- Средний рейтинг конкретного товара

1. SELECT AVG(Rating), products.`Name`, products.Model FROM reviews

JOIN products ON Product\_ID = products.ID

JOIN categories ON Category\_ID = categories.ID

WHERE products.`Name` = 'Xiaomi Mi TV';



-- Средний рейтинг всех товаров компании

1. SELECT AVG(Rating), products.`Name` FROM reviews

JOIN products ON Product\_ID = products.ID

WHERE products.`Name` LIKE 'Apple%';



# Индексы, хранимые процедуры, представления

1. -- Общая стоимость всех товаров, проданных конкретным сотрудником

SELECT SUM(Price\_per\_unit \* instances.Amount), emploees.Fullname FROM instances\_in\_orders

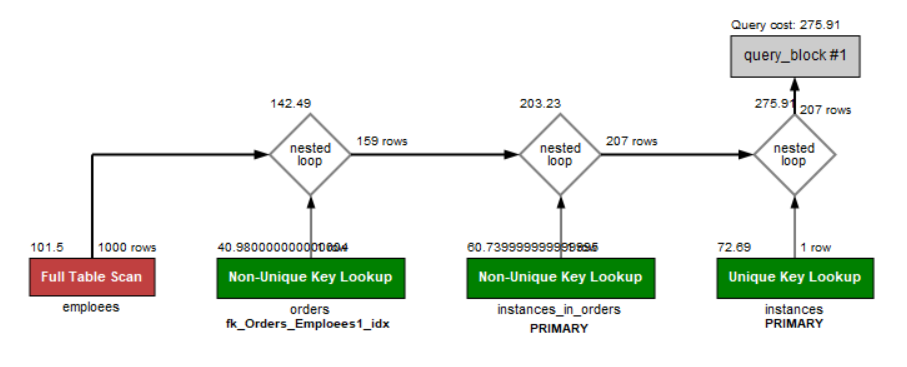
JOIN instances ON Instance\_product\_ID = instances.Product\_ID AND Instance\_shop\_ID = instances.Shop\_ID

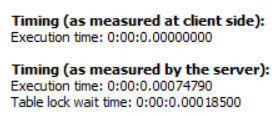
JOIN orders ON Order\_ID = orders.ID

JOIN emploees ON Emploees\_ID = emploees.ID

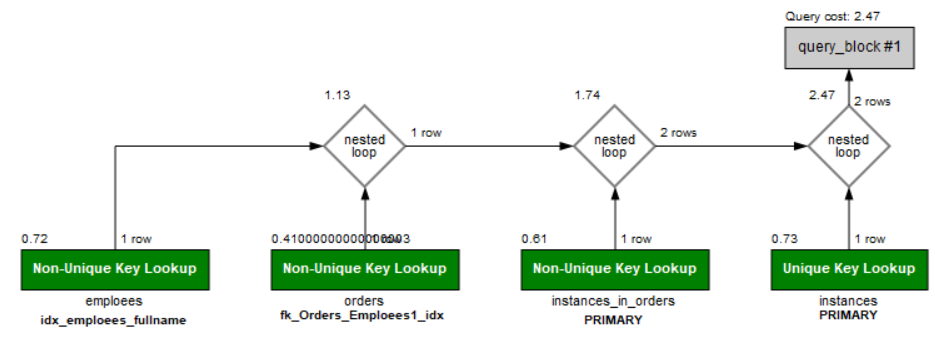
WHERE emploees.Fullname = 'Haven Nast';

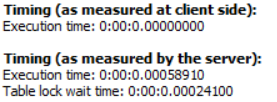
Без индекса:





После создания индекса:





1. -- Общее количество денег, потраченных конкретным клиентом

SELECT SUM(Price\_per\_unit \* instances.Amount), clients.Fullname FROM instances\_in\_orders

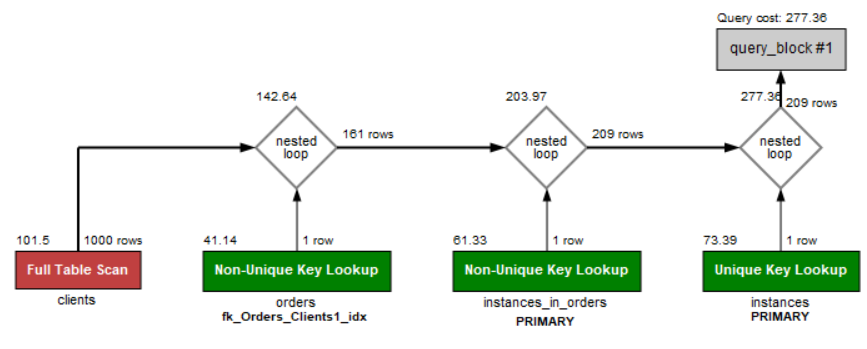
JOIN instances ON Instance\_product\_ID = instances.Product\_ID AND Instance\_shop\_ID = instances.Shop\_ID

JOIN orders ON Order\_ID = orders.ID

JOIN clients ON Client\_ID = clients.ID

WHERE clients.Fullname = 'Laural Fawke';

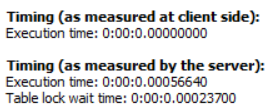
Без индекса:





После создания индекса:





1. -- Все отзывы о конкретном товаре

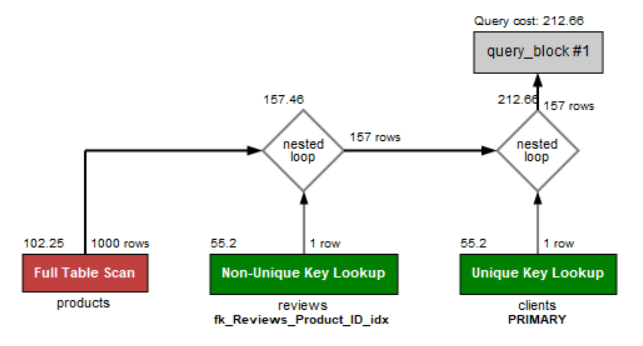
SELECT products.`Name`, products.Model, reviews.Rating, reviews.`Description` FROM products

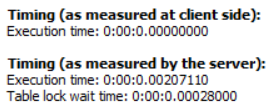
JOIN reviews ON products.ID = reviews.Product\_ID

JOIN clients ON reviews.Client\_ID = clients.ID

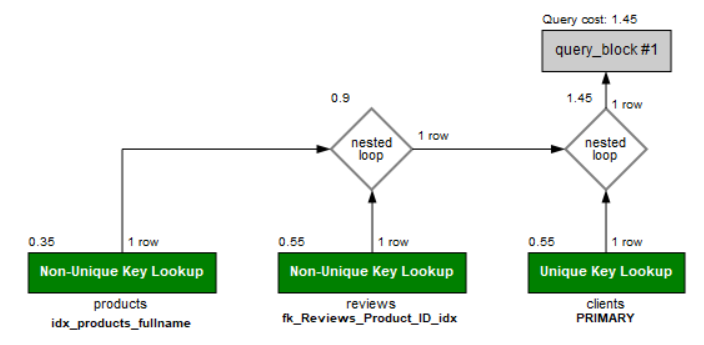
WHERE products.`Name` = "Crucial Ballistix";

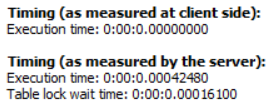
Без индекса:





После создания индекса:





1. -- Средний рейтинг конкретного товара

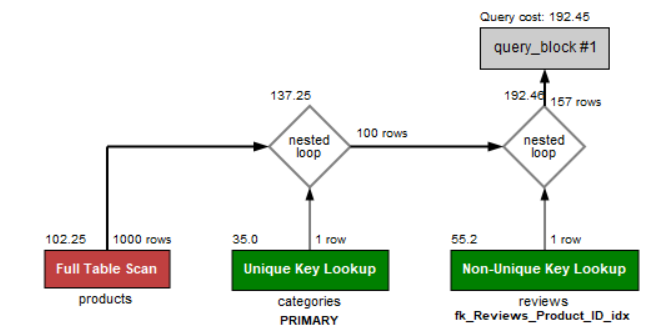
SELECT AVG(Rating), products.`Name`, products.Model FROM reviews

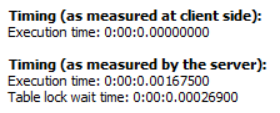
JOIN products ON Product\_ID = products.ID

JOIN categories ON Category\_ID = categories.ID

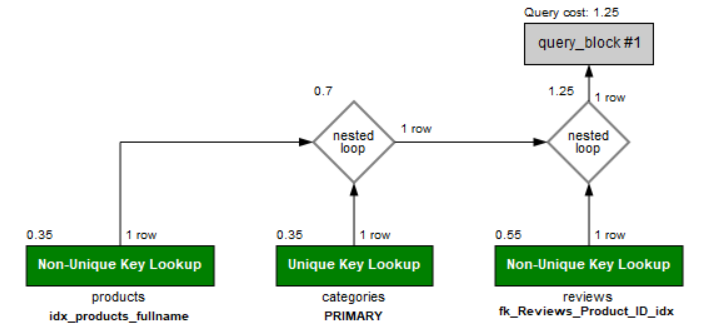
WHERE products.`Name` = 'Xiaomi Mi TV';

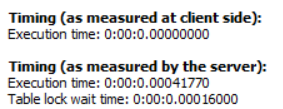
Без индекса:





После создания индекса:





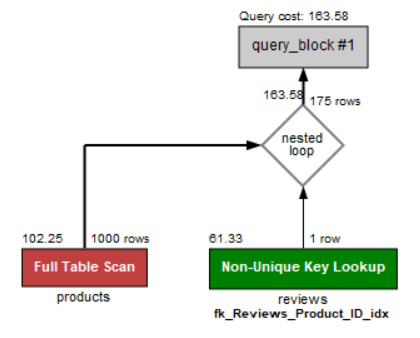
1. -- Средний рейтинг всех товаров компании

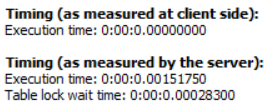
SELECT AVG(Rating), products.`Name` FROM reviews

JOIN products ON Product\_ID = products.ID

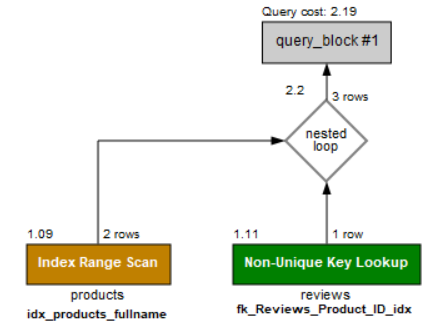
WHERE products.`Name` LIKE 'Apple%';

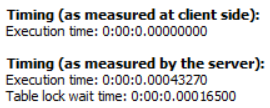
Без индекса:





После создания индекса:





**Процедуры**:

1. -- Сумма всех затрат клиента

DELIMITER $$

CREATE PROCEDURE customer\_costs(client\_fullname VARCHAR(255))

BEGIN

SELECT SUM(Price\_per\_unit \* instances.Amount), clients.Fullname FROM instances\_in\_orders

JOIN instances ON Instance\_product\_ID = instances.Product\_ID AND Instance\_shop\_ID = instances.Shop\_ID

JOIN orders ON Order\_ID = orders.ID

JOIN clients ON Client\_ID = clients.ID

WHERE clients.Fullname = client\_fullname;

END$$



1. -- Все отзывы о конкретном товаре

DELIMITER $$

CREATE PROCEDURE product\_reviews(product\_name VARCHAR(255))

BEGIN

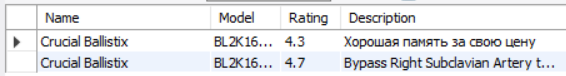
SELECT products.`Name`, products.Model, reviews.Rating, reviews.`Description` FROM products

JOIN reviews ON products.ID = reviews.Product\_ID

JOIN clients ON reviews.Client\_ID = clients.ID

WHERE products.`Name` = product\_name;

END$$



1. -- Средний рейтинг конкретного товара

DELIMITER $$

CREATE PROCEDURE product\_rating(product\_id VARCHAR(255))

BEGIN

SELECT AVG(Rating), products.`Name`, products.Model FROM reviews

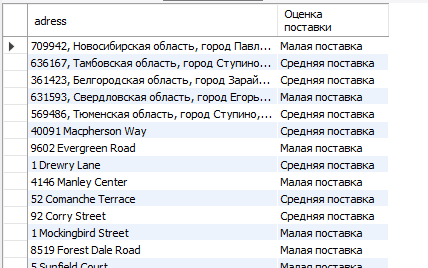
JOIN products ON Product\_ID = products.ID

JOIN categories ON Category\_ID = categories.ID

WHERE products.ID = product\_id;

END$$





**Функции**:

1. -- Найти ID товара по его названию и модели

DELIMITER $$

CREATE FUNCTION get\_product\_id(product\_name VARCHAR(255), product\_model VARCHAR(255))

RETURNS INT

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE id INT DEFAULT 0;

SELECT products.ID INTO id FROM products

WHERE `Name` = product\_name AND Model = product\_model;

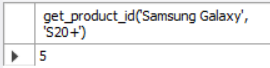
IF(id > 0) THEN

RETURN id;

END IF;

RETURN 0;

END$$



1. -- Найти минимальную цену товара

DELIMITER $$

CREATE FUNCTION min\_product\_cost(\_product\_id INT)

RETURNS INT

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE cost INT DEFAULT 0;

SELECT MIN(instances.Price\_per\_unit) INTO cost FROM instances WHERE Product\_ID = \_product\_id;

IF(\_product\_id > 0) THEN

RETURN cost;

END IF;

RETURN 0;

END$$



1. -- Проверить наличие товара

DELIMITER $$

CREATE FUNCTION check\_availability(productID INT, city VARCHAR(128))

RETURNS BOOL

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE count INT DEFAULT 0;

IF(productID < 1) THEN

RETURN FALSE;

END IF;

SELECT COUNT(\*) INTO count FROM instances

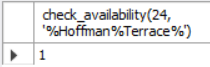
WHERE EXISTS(SELECT \* FROM instances

JOIN shops ON Shop\_ID = shops.ID

WHERE Product\_ID = productID AND Address LIKE city);

RETURN count > 0;

END$$



**Представления**

-- Средний рейтинг категорий

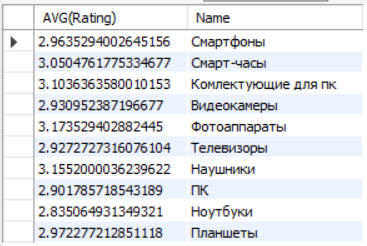
CREATE VIEW categories\_avg\_rating AS

SELECT AVG(Rating), categories.Name FROM reviews

JOIN products ON Product\_ID = products.ID

JOIN categories ON Category\_ID = categories.ID

GROUP BY Category\_ID;



2) -- Все товары в наличии

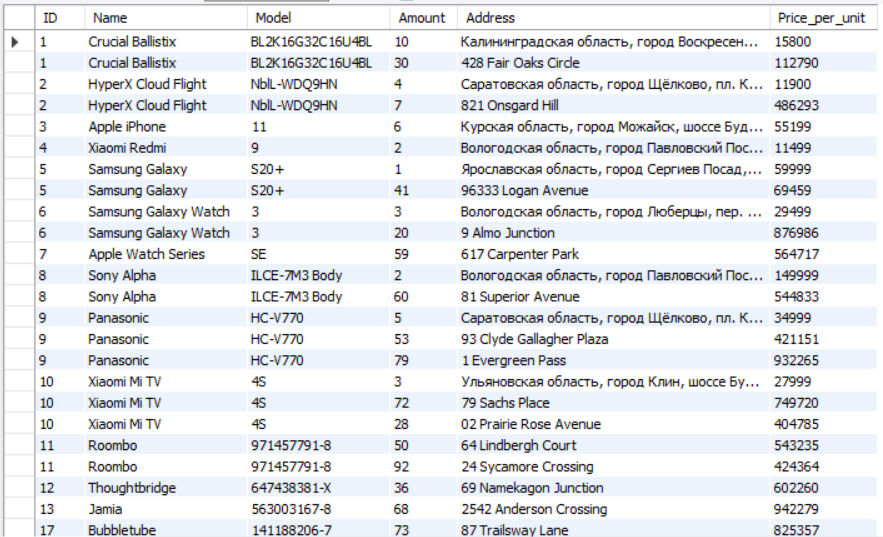
CREATE VIEW available\_products AS

SELECT products.ID, products.`Name`, products.Model, instances.Amount, shops.Address, instances.Price\_per\_unit FROM products

JOIN instances ON products.ID = instances.Product\_ID

JOIN shops ON instances.Shop\_ID = shops.ID

WHERE instances.Amount > 0;



3) -- Актуальные заказы

CREATE VIEW current\_instances\_in\_otders AS

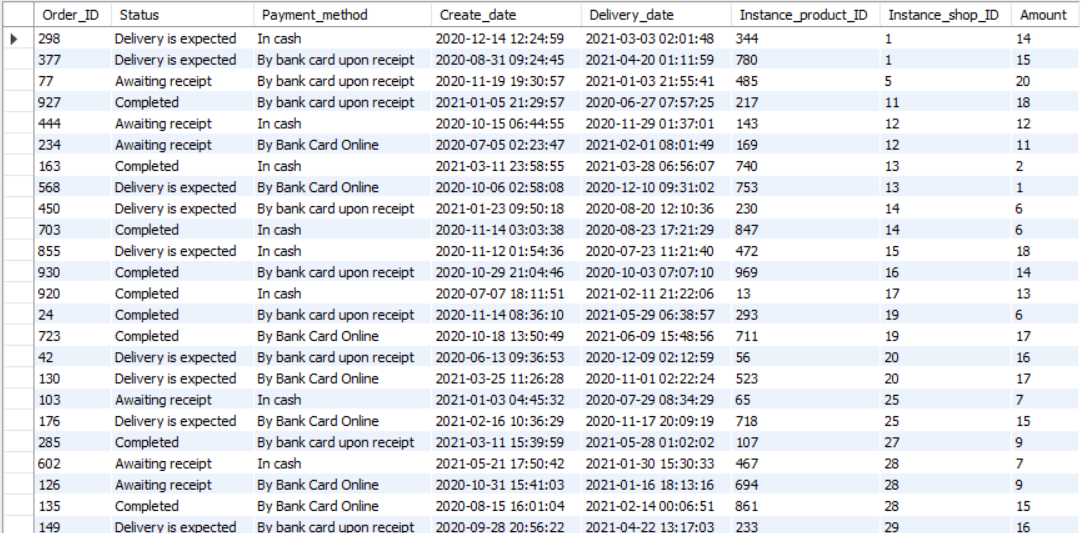
SELECT Order\_ID, `Status`, Payment\_method, Create\_date, Delivery\_date, Instance\_product\_ID, Instance\_shop\_ID, instances\_in\_orders.Amount

FROM instances

JOIN instances\_in\_orders

ON instances.Product\_ID = instances\_in\_orders.Instance\_product\_ID AND instances.Shop\_ID = instances\_in\_orders.Instance\_shop\_ID

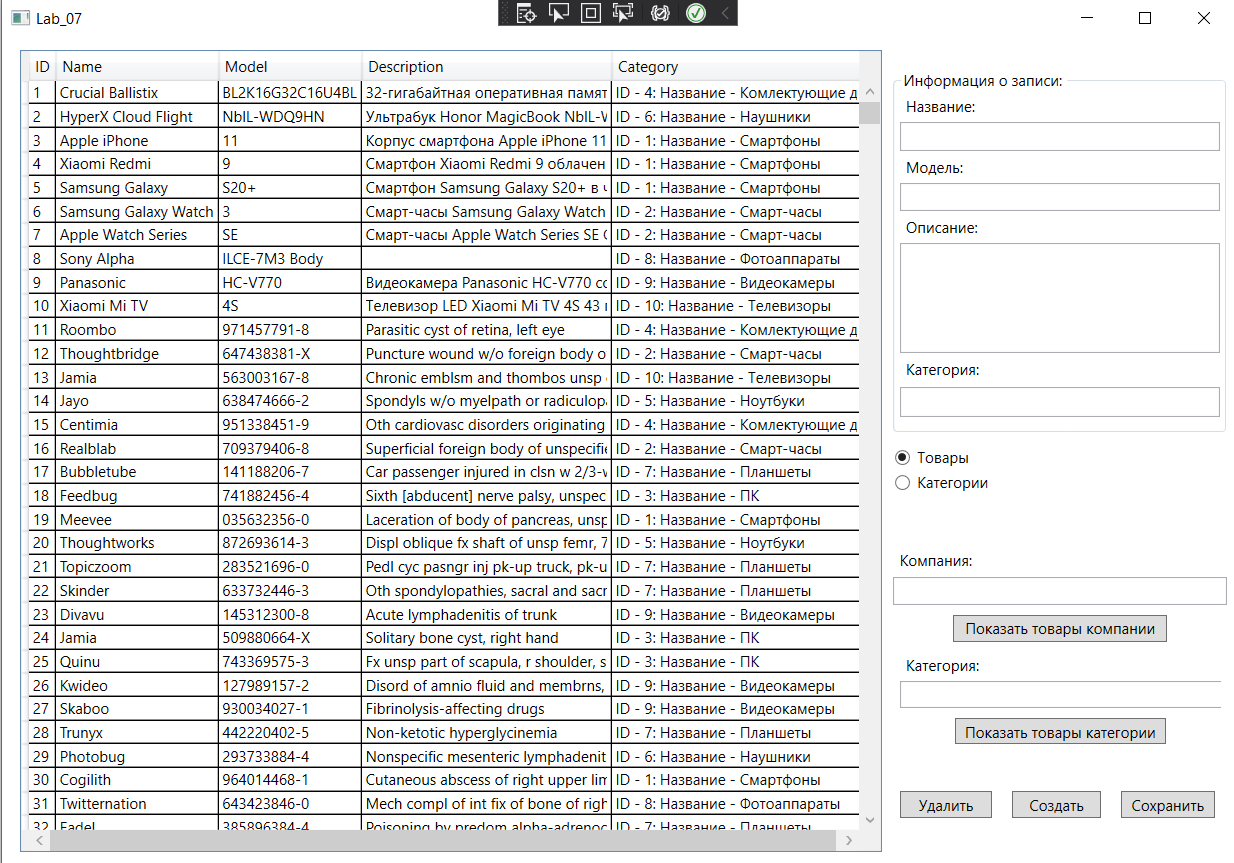
JOIN orders ON instances\_in\_orders.Order\_ID = orders.ID;



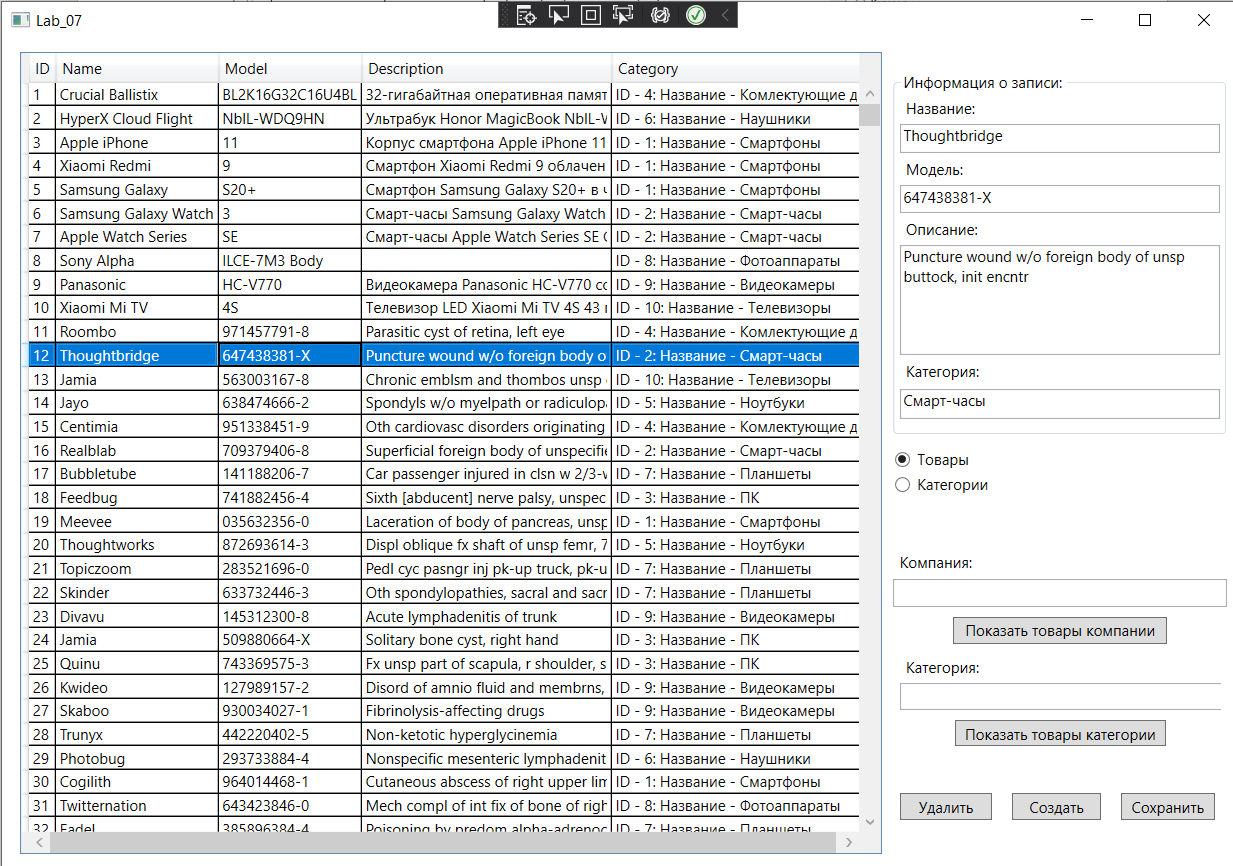
# Взаимодействие с БД из приложения

1. Для реализации лабораторной работы, а также курсовой работы был выбран язык программирования C#.
2. Были изучены библиотеки для работы с выбранной СУБД и на их основе было реализовано простейшее приложение с графическим интерфейсом для подключения к собственной БД в СУБД на выбранном языке программирования использовалась библиотека MySql.Data.MySqlClient.
3. В качестве интерфейса взаимодействия с пользователем на данной лабораторной работе было реализовано оконное приложение (WPF).
4. Исходный код загружен в репозиторий, ссылка на который находится в протоколе.
5. Была реализована CRUD (получение, добавление, редактирование и удаление данных) для 2-х таблиц базы данных, одна из которых связана внешним ключом с другой.
6. Была протестирована логика взаимодействия с БД, с помощью проверки данных в БД можно убедиться, что все функции приложения выполняются корректно. Методы, в которых содержатся запросы к БД, возвращают корректные данные, корректно производится запись, создаются корректные связи между сущностями БД.

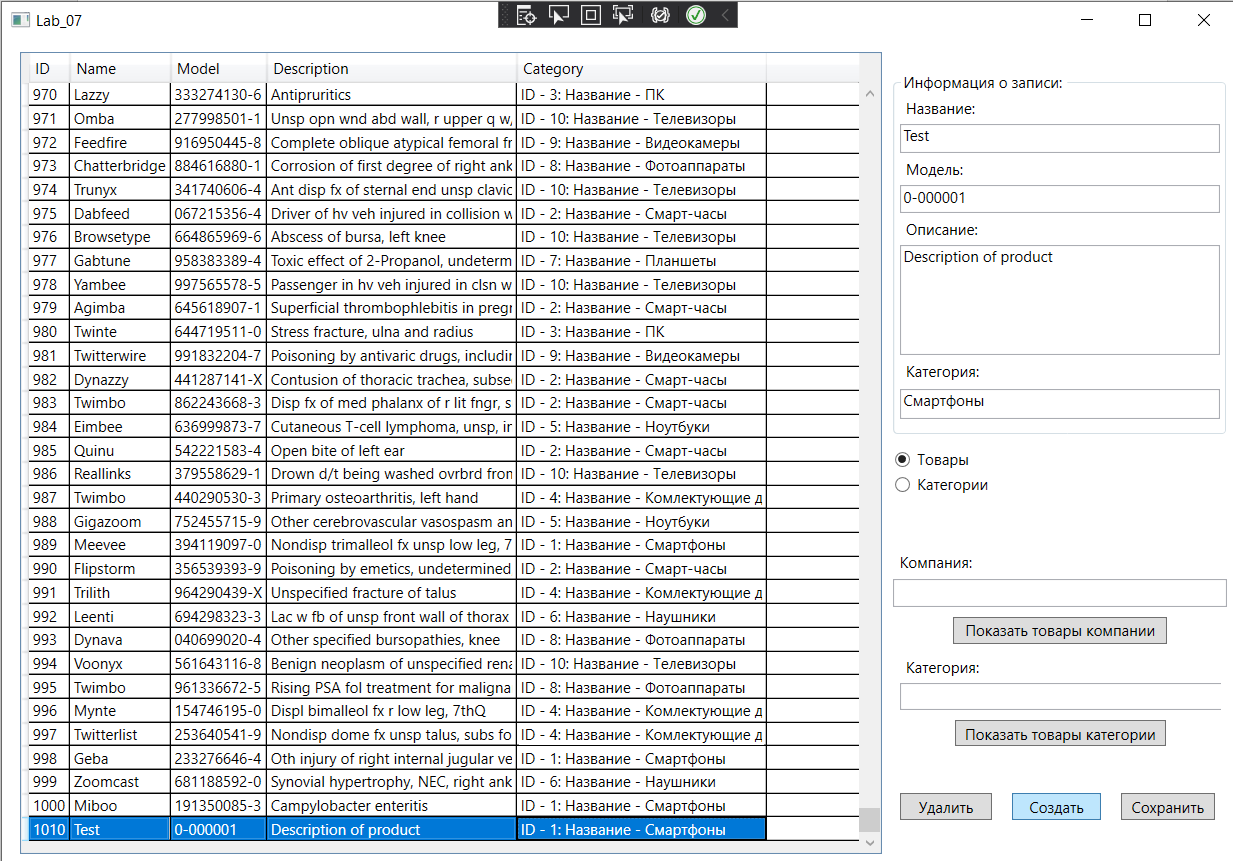
Результаты работы приложения:



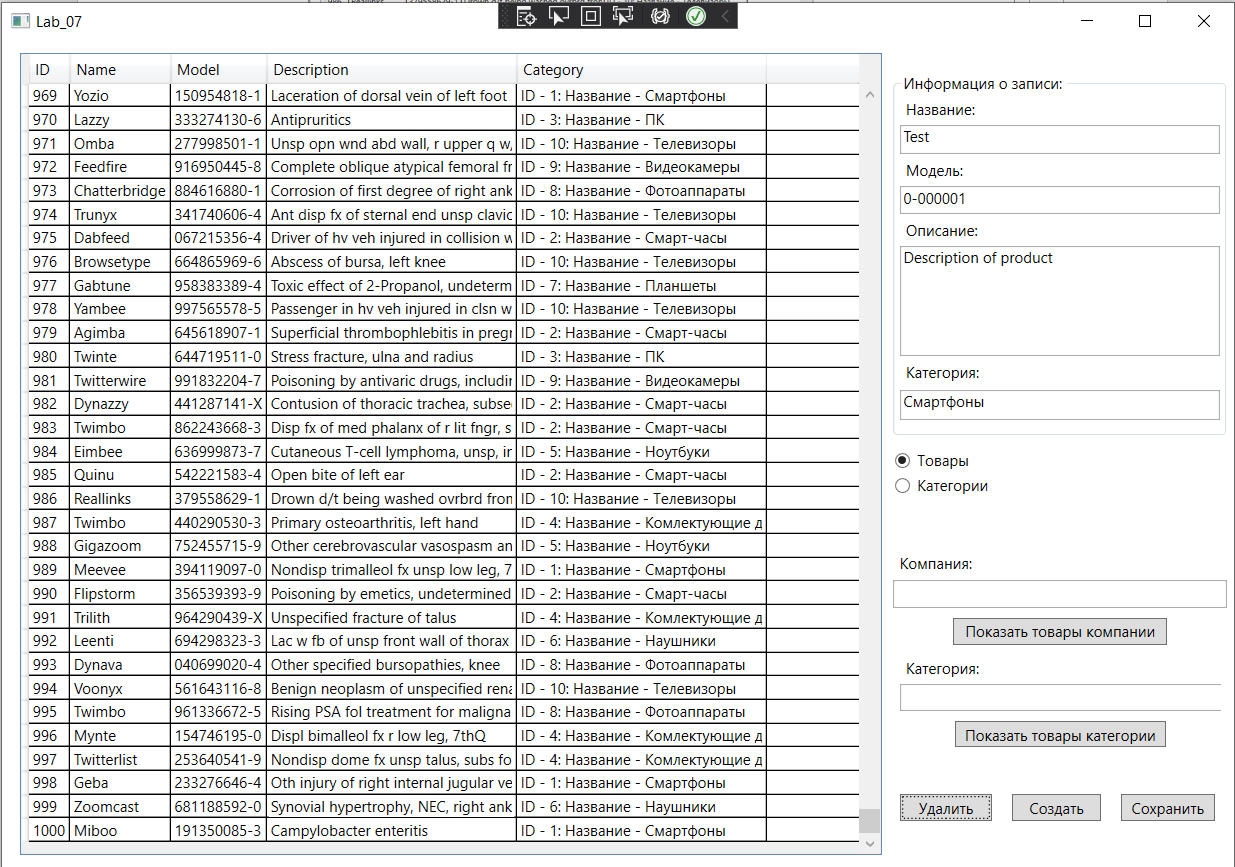
Загрузка данных из базы:



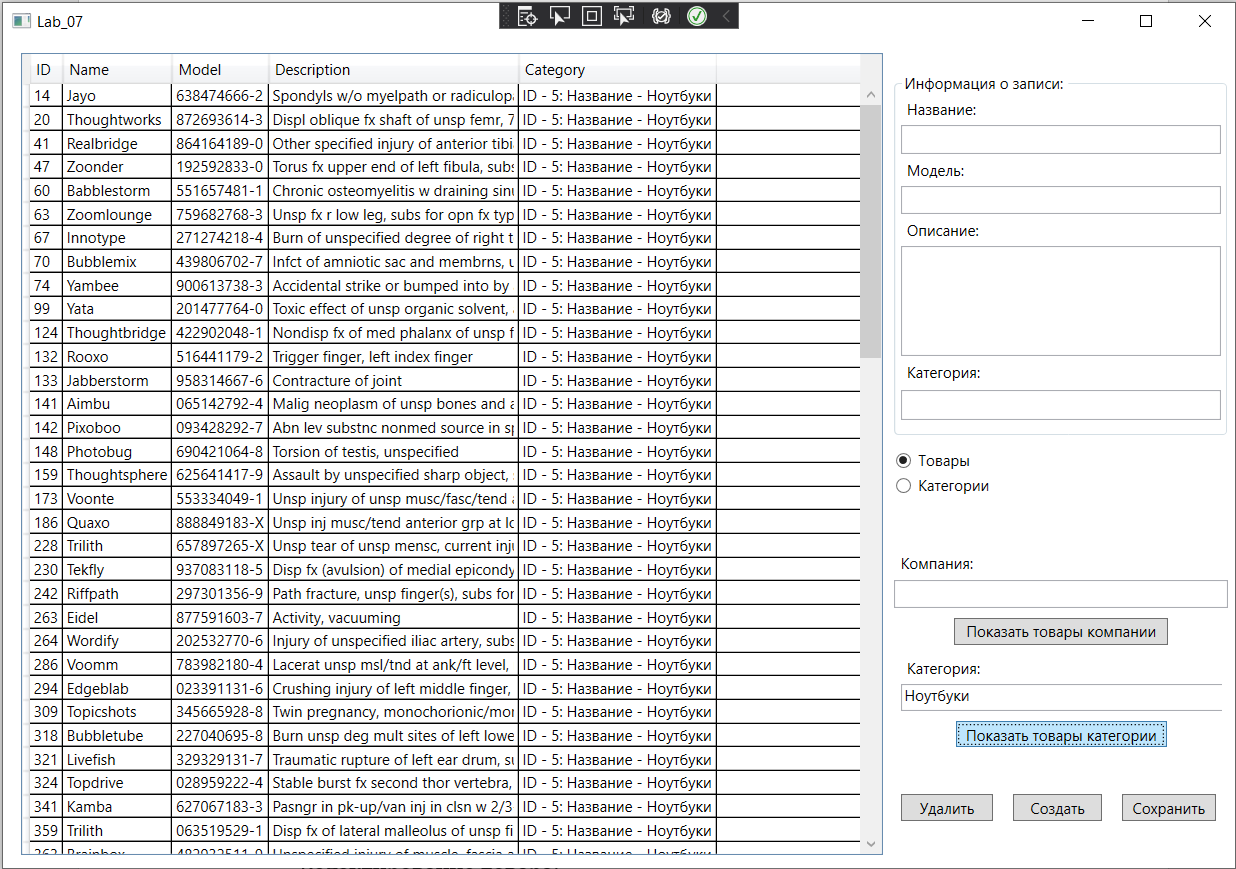
Добавление товара в базу:



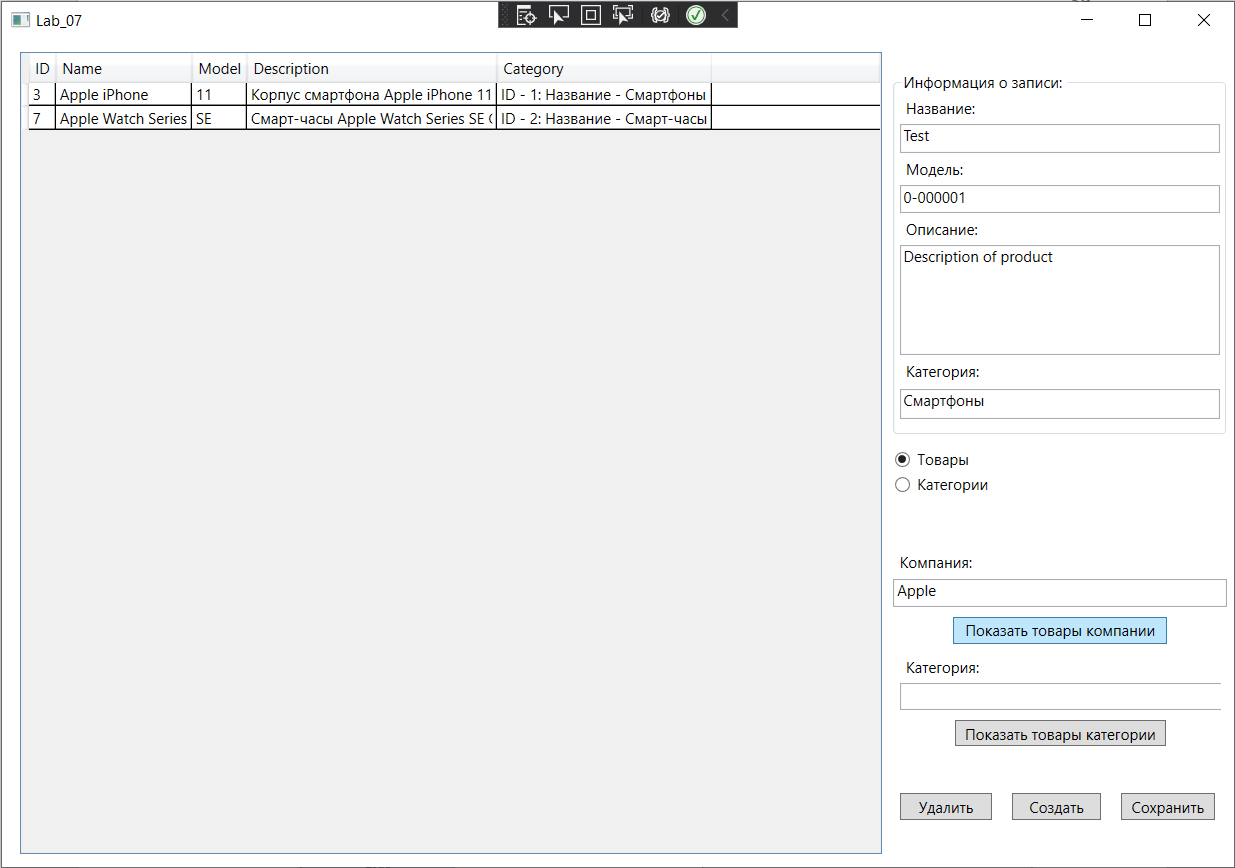
Удаление товара из базы:



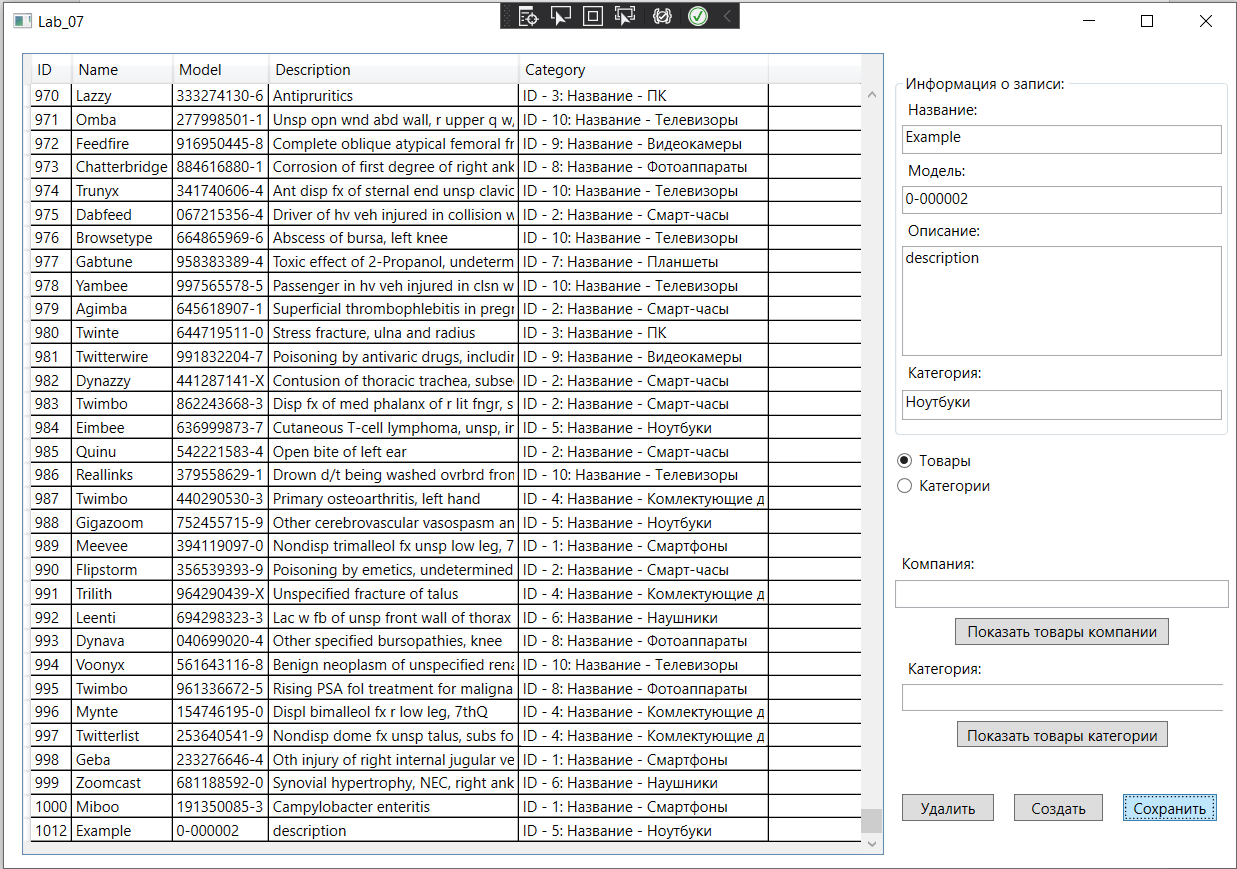
Товары из заданной категории:



Товары заданной компании:



Редактирование товара:



**Вывод:**

В ходе работы была разработана программа на C# для работы с базой данных на SQL server.

**Заключение:**

В ходе работы была разработана CУБД «Интернет-магазин электроники» в среде разработки MySQL Workbench и программа на языке C# для непосредственной работы с ней.

# Ссылка на репозиторий:

<https://github.com/Tapo4ek-exe/electronics_shop_db/tree/Lab7>

# Список литературы

1. Учебно-методическое пособие БД
2. Интернет-статья https://wiki.gentoo.org/wiki/MySQL/Startup\_Guide/ru