**Міністерство оствіти та науки України**

**Дніпропетровський національний університет залізничного транспорту**

**імені академіка В.Лазаряна**

**КУРСОВА РОБОТА**

Розробка реляційної бази даних, проектування структури сховища даних

з навчальної дисципліни «Бази даних»

Виконав:

студент групи 941

Валовый Б. Ю.

Науковий керівник:

Пюра А. Н.

# Дніпро 2018

# 1 Постановка задачи, выбор предметной области, спецификация требований к управлению данными, определение основных типов запросов

Предметная область – модель интернет-магазина аксесуаров для ноутбуков и телефонов.

## Разработка информационно-логической и концептуальной EER модели БД

На рисунке 1.1 изображена концептуальная диаграмма модели БД.

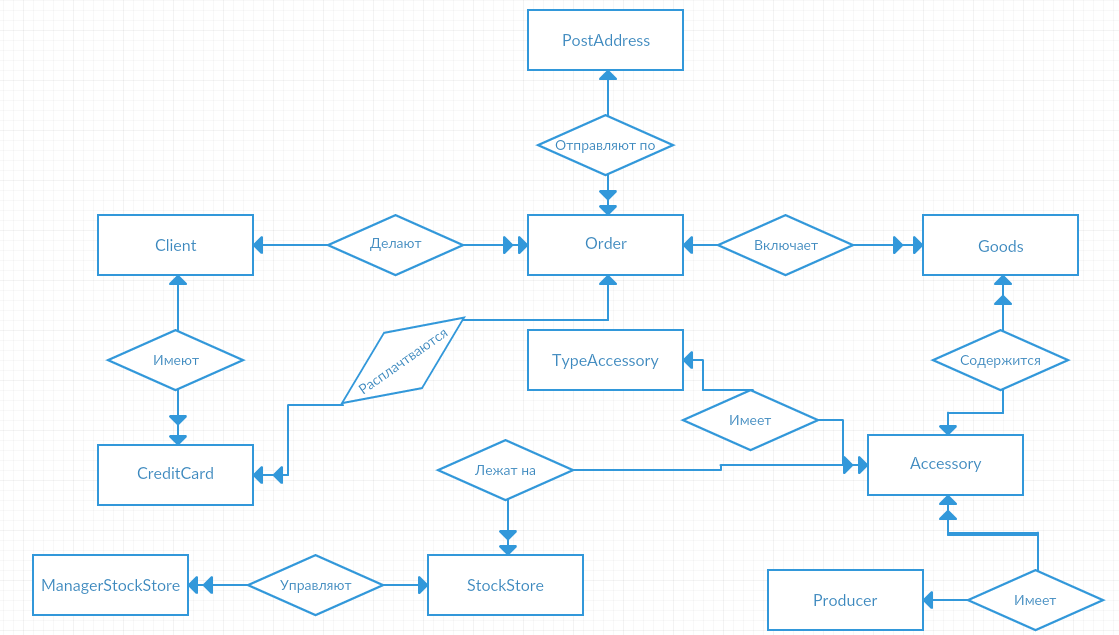


Рис. 1.1 – Концептуальная диаграмма модели БД

## 1.2 Определение типов сущностей, атрибутов, характеристик связей, установления возможных и первичных ключей

PK – primary key; FK – foreign key

**Client:**

* Id(primary key)
* First Name
* Last Name
* Phone number

**Order:**

* Id(PK)
* Id client(FK)
* Id post address(FK)

**Goods:**

* Id accessory(FK)
* Id order(FK)
* Accessory count

**Post address:**

* Id(PK)
* City
* Street
* Number of post department

**Producer:**

* Id(PK)
* Company name
* Country

**Accessory type:**

* Id(PK)
* Type name

**Accessory:**

* Id(PK)
* Name
* Color
* Type of link
* Id producer(FK)
* Id type(FK)

**StockStore:**

* StockId(PK)
* AccessoryTypeId(FK)
* City
* AddressOfStock

**StockStoreAccessory:**

* StockId(FK)
* AccessoryId(FK)
* PrimaryKey(StockId, AccessoryId)

**CreditCard:**

* CCNumb(PK)
* FirstName
* LastName
* ExpiryDate
* ValidationCode
* ClientId(FK)
* OrderId(FK)

**ManagerStockStore:**

* ManagerId(PK)
* FirstName
* LastName
* Phone
* StockId(FK)

## 1.3 Анализ требований к БД, выполнение процедур специализации / генерации, категоризации типов сущностей EER модели БД

Правильно спроектированная БД должна удовлетворять следующим требованиям:

1. Минимальная избыточность. Непротиворечивость.

2. Целостность данных.

3. Независимость данных.

4. Возможность ведения (добавления и удаления) и актуализации (корректировки, модификации) данных.

5. Безопасность и секретность.

6. Высокая производительность. Минимальные затраты.

В разработанной базе данных были выделены следующей типы сущностей:

1. Client
2. Order
3. Goods
4. Post address
5. Producer
6. Accessory type
7. Accessory
8. CreditCard
9. StockStore
10. ManagerStockStore
11. StockStoreAccessory

# Разработка логического проекта базы данных

## 2.1 Разработка предварительной логической схемы БД на основе EER модели

На рисунке 1.2 изображена логическая схема БД на основе EER модели.



Рис. 1.2 – Логическая схема БД на основе EER модели

## 2.2 Распределение не ключевых атрибутов, нормализация БД.

Описание процесса нормализации отношений базы данных: база данных была приведена к третей форме. В сущности Order были найдены транзитивные зависимости City -> Street -> Number of post и успешно нормализованы. На рисунке 1.3 изображена нормализованная логическая схема БД на основе EER модели



Рис. 1.3 – Нормализованная логическая схема БД на основе EER модели

## 2.3 Удаление лишних, сложных (m : n) и рекурсивных связей, а также связей, которые имеют атрибуты

В ходе проектирования логической схемы базы данных было обнаружено, что логическая схема БД не содержит лишних, сложных или рекурсивных связей, от которых необходимо избавляться.

## 2.4 Обеспечение целостности БД

Целосность БД была обеспечена путем добавления к атрибутам таких ограничений: primary key, foreign key, check, , то есть можно выделить следующие типы ограничения целостности БД:

* Ограничение атрибутов в отношениях (ограничение на значения, которые может принимать указанный атрибут).
* Ограничение типа (определение множества значений, из которых состоит данный тип).
* Ограничение на пустоту поля (большинство полей базы данных не может содержать значение NULL).

## 2.5 Анализ запросов, создание карт транзакций

Чтобы клиент мог оперировать данными с базы данных необходимо создать запросы, которые позволят ему делать это. Поэтому определим запросы, которые будут полезны клиенту для работы с БД:

* Запрос, отображающий информацию о клиентах интернет магазина.
* Запрос, отображающий информацию о акссесуарах на складах интернет магазина.
* Запрос, отображающий информацию о заказах клиента интернет-магазина.
* Запрос, отображающий информацию о текущих складах интернет-магазина.
* Запросы, которые позволяют добавлять новых клиентов.
* Запросы, которые позволяют добавлять новые заказы.
* Запросы, которы позволяют добавлять новые аксессуары.
* Запросы, которые позволяют изменять текущие аксессуары.
* Запросы, которые позволяют изменять клиентов.
* Запросы, которые позволяют удалять клиентов.
* Запросы, которые позволяют удалять аксессуары.

# Разработка физического проекта базы данных

## Разработка структур таблиц и схемы БД с учетом возможностей СУБД. Скрипт создания БД

/\* Создание Базы Данных\*/

-------------------------------------------------------------------------

CREATE DATABASE ShopDB

ON

(

NAME = 'ShopDB',

FILENAME = 'G:\Studying\4\_course\DataBase\CourseWorkDB\DB\ShopDB.mdf',

SIZE = 200MB,

MAXSIZE = 1000MB,

FILEGROWTH = 100MB

)

LOG ON

(

NAME = 'LogShopDB',

FILENAME = 'G:\Studying\4\_course\DataBase\CourseWorkDB\DB\LogShopDB.ldf',

SIZE = 100MB,

MAXSIZE = 500MB,

FILEGROWTH = 50MB

)

COLLATE Cyrillic\_General\_CI\_AS -- Задаем кодировку для базы данных по умолчанию

-------------------------------------------------------------------------

-- Используем базу данных интернет-магазина

USE ShopDB

GO

-------------------------------------------------------------------------

/\* Создание таблиц \*/

-------------------------------------------------------------------------

CREATE SCHEMA Shop

GO

--таблица клиентов

CREATE TABLE Shop.Client

(

ClientId int IDENTITY(1,1) NOT NULL,

FirstName nvarchar(20) NOT NULL,

LastName nvarchar(20) NOT NULL,

Phone char(10) UNIQUE

)

-- Создание ограничения на телефон

ALTER TABLE Shop.Client

ADD CONSTRAINT CN\_ClientsPhone

-- Ограничение CHECK

CHECK (Phone LIKE '0[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

GO

-- Изменяем таблицу Client задав ограничение первичного ключа на столбце ClientId

ALTER TABLE Shop.Client

ADD CONSTRAINT PK\_Client

PRIMARY KEY (ClientId)

GO

--таблица заказов

CREATE TABLE Shop.Orders

(

OrderId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

ClientId int NOT NULL FOREIGN KEY REFERENCES Shop.Client(ClientId)

ON DELETE CASCADE ON UPDATE CASCADE,

City nvarchar(30) NOT NULL,

Street nvarchar(30) NOT NULL,

NumberOfStreet nvarchar(10) NOT NULL

)

--таблица карточки для виплат клиента

CREATE TABLE Shop.CreditCard

(

CCNumb bigint NOT NULL PRIMARY KEY,

FirstName nvarchar(20) NOT NULL,

LastName nvarchar(20) NOT NULL,

ExpiryDate Date NOT NULL,

ClientId int NOT NULL FOREIGN KEY REFERENCES Shop.Client(ClientId)

ON DELETE CASCADE ON UPDATE CASCADE,

OrderId int NULL FOREIGN KEY REFERENCES Shop.Orders(OrderId)

ON DELETE SET NULL ON UPDATE CASCADE

)

--производитель товаров

CREATE TABLE Shop.Producer

(

ProducerId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

CompanyName nvarchar(20),

Country nvarchar(20)

)

GO

--тип аксесуара

CREATE TABLE Shop.AccessoryType

(

AccessoryTypeId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

TypeName nvarchar(20) NOT NULL

)

--склады для аксесуара

CREATE TABLE Shop.StockStore

(

StockId int IDENTITY(1,1) NOT NULL PRIMARY KEY,

City nvarchar(30) NOT NULL,

AddressOfStock nvarchar(30) NOT NULL

)

--аксесуар

CREATE TABLE Shop.Accessory

(

AccessoryId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

AccessoryName nvarchar(60) NOT NULL,

Color nvarchar(30) NOT NULL,

TypeOfLink nvarchar(30) NOT NULL,

ProducerId int NOT NULL

FOREIGN KEY REFERENCES Shop.Producer(ProducerId),

AccessoryTypeId int NOT NULL

FOREIGN KEY REFERENCES Shop.AccessoryType(AccessoryTypeId)

)

GO

--многие ко многим StockStore - Accessory

CREATE TABLE Shop.StockStoreAccessory

(

StockId int NOT NULL

FOREIGN KEY REFERENCES Shop.StockStore(StockId),

AccessoryId int NOT NULL

FOREIGN KEY REFERENCES Shop.Accessory(AccessoryId)

ON DELETE CASCADE ON UPDATE CASCADE,

PRIMARY KEY(StockId, AccessoryId)

)

GO

--менеджер склада

CREATE TABLE Shop.ManagerStockStore

(

ManagerId int IDENTITY(1,1) NOT NULL PRIMARY KEY,

FirstName nvarchar(20) NOT NULL,

LastName nvarchar(20) NOT NULL,

Phone char(10) UNIQUE,

StockId int NOT NULL

)

-- Создание ограничения на телефон

ALTER TABLE Shop.ManagerStockStore

ADD CONSTRAINT CN\_ManagerPhone

-- Ограничение CHECK

CHECK (Phone LIKE '0[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

GO

ALTER TABLE Shop.ManagerStockStore

ADD CONSTRAINT FK\_StockStoreId

FOREIGN KEY (StockId) REFERENCES Shop.StockStore(StockId)

GO

--товар

CREATE TABLE Shop.Goods

(

OrderId int NOT NULL,

AccessoryId int NOT NULL

FOREIGN KEY REFERENCES Shop.Accessory(AccessoryId),

CountAccessory smallint NOT NULL

)

GO

ALTER TABLE Shop.Goods

ADD CONSTRAINT FK\_OrderId

FOREIGN KEY (OrderId) REFERENCES Shop.Orders(OrderId)

ON DELETE CASCADE ON UPDATE CASCADE

GO

## 3.2. Разработка параметризованных запросов с использованием объединения таблиц.

--Выберем все акссесуары

CREATE FUNCTION Accessories()

RETURNS TABLE

AS

RETURN SELECT

Access.AccessoryId AS [Id],

AccessType.TypeName AS [Тип аксессуара],

Access.AccessoryName AS [Название аксессуара],

Access.Color AS [Цвет],

Access.TypeOfLink AS [Тип соединения],

Produc.CompanyName AS [Название компании],

Produc.Country AS [Страна производитель]

FROM Shop.Accessory AS Access

INNER JOIN Shop.AccessoryType AS AccessType

ON AccessType.AccessoryTypeId = Access.AccessoryTypeId

INNER JOIN Shop.Producer AS Produc

ON Produc.ProducerId = Access.ProducerId

GO

--выберем все типы аксессуаров

CREATE FUNCTION AccessoryTypes()

RETURNS TABLE

AS

RETURN

SELECT TypeName FROM Shop.AccessoryType

GO

--выберем всех производителей

CREATE FUNCTION Producers()

RETURNS TABLE

AS

RETURN

SELECT CompanyName, Country FROM Shop.Producer

GO

-- выберем всех клиентов

CREATE FUNCTION Clients()

RETURNS TABLE

AS

RETURN SELECT

Shop.Client.ClientId AS [Id клиента],

Shop.Client.FirstName AS [Имя клиента],

Shop.Client.LastName AS [Фамилия клиента],

Shop.Client.Phone AS [Телефон клиента]

FROM Shop.Client

GO

--веберем все заказы

CREATE FUNCTION Orders()

RETURNS TABLE

AS

RETURN

SELECT Shop.Orders.OrderId AS [Id заказа],

Shop.Client.FirstName AS [Имя клиента],

Shop.Client.LastName AS [Фамилия клиента],

Shop.Client.Phone AS [Номер телефона],

Shop.Accessory.AccessoryName AS [Аксессуар],

Shop.Accessory.Color AS [Цвет],

Shop.Accessory.TypeOfLink AS [Тип соединения],

Shop.Goods.CountAccessory AS [Количество]

FROM Shop.Orders

INNER JOIN Shop.Client

ON Shop.Client.ClientId = Shop.Orders.ClientId

INNER JOIN Shop.Goods

ON Shop.Orders.OrderId = Shop.Goods.OrderId

INNER JOIN Shop.Accessory

ON Shop.Accessory.AccessoryId = Shop.Goods.AccessoryId

GO

--выберем склады

CREATE FUNCTION StockStores()

RETURNS TABLE

AS

RETURN SELECT

Shop.StockStore.City AS [Город склада],

Shop.StockStore.AddressOfStock AS [Улица склада],

Shop.ManagerStockStore.LastName AS [Фамилия управляющего складом],

Shop.ManagerStockStore.FirstName AS [Имя управляющего складом],

Shop.ManagerStockStore.Phone AS [Телефон управляющего складом]

FROM Shop.StockStore

INNER JOIN Shop.ManagerStockStore

ON Shop.ManagerStockStore.StockId = Shop.StockStore.StockId

GO

--добавить клиента

CREATE PROC InsertClient

@FName nvarchar(20),

@LName nvarchar(20),

@Phone nvarchar(20)

AS

BEGIN

INSERT INTO Shop.Client

VALUES

(@FName, @LName, @Phone);

RETURN SCOPE\_IDENTITY(); -- id

END

GO

DROP PROC InsertClient

GO;

-- добавить аксессуар

CREATE PROC InsertAccessory

@AccessName nvarchar(60),

@Color nvarchar(30),

@TypeOfLink nvarchar(30),

@ProducerId int,

@AccessTypeId int

AS

BEGIN

INSERT INTO Shop.Accessory

VALUES

(@AccessName, @Color, @TypeOfLink, @ProducerId, @AccessTypeId);

END

GO

--добавить заказ

CREATE PROC InsertOrder

@ClientId int,

@City nvarchar(30),

@Street nvarchar(30),

@NumberOfStreet nvarchar(10)

AS

BEGIN

INSERT INTO Shop.Orders

VALUES

(@ClientId, @City, @Street, @NumberOfStreet);

RETURN SCOPE\_IDENTITY(); -- id

END

GO

--добавить товар заказа

CREATE PROC InsertGoods

@OrderId int,

@AccessoryId int,

@CountAccessory smallint

AS

BEGIN

INSERT INTO Shop.Goods

VALUES

(@OrderId, @AccessoryId, @CountAccessory);

END

GO

--удаление клиента

CREATE PROC DeleteClient

@ClientId int

AS

BEGIN

DELETE Shop.Client

WHERE Shop.Client.ClientId = @ClientId;

END

GO

--удаление аксессуара

CREATE PROC DeleteAccessory

@AccessId int

AS

BEGIN

DELETE Shop.Accessory

WHERE Shop.Accessory.AccessoryId = @AccessId;

END

GO

--обновить клиента

CREATE PROC UpdateClient

@ClientId int,

@FName nvarchar(20),

@LName nvarchar(20),

@Phone nvarchar(20)

AS

BEGIN

UPDATE Shop.Client

SET FirstName = @FName, LastName = @LName, Phone = @PHone

WHERE ClientId = @ClientId

END

GO

-- обновить аксессуар

CREATE PROC UpdateAccessory

@AccessId int,

@AccessName nvarchar(60),

@Color nvarchar(30),

@TypeOfLink nvarchar(30)

AS

BEGIN

UPDATE Shop.Accessory

SET AccessoryName = @AccessName,

Color = @Color,

TypeOfLink = @TypeOfLink

WHERE Shop.Accessory.AccessoryId = @AccessId

END

GO

## 3.3. Разработка клиента БД: формы редактирования данных таблиц, формы параметризованных запросов.

Формы выборки данных из базы данных. На рисунках 1.4, 1.5, 1.6, 1.7 изображены формы выборки данных из таблиц:

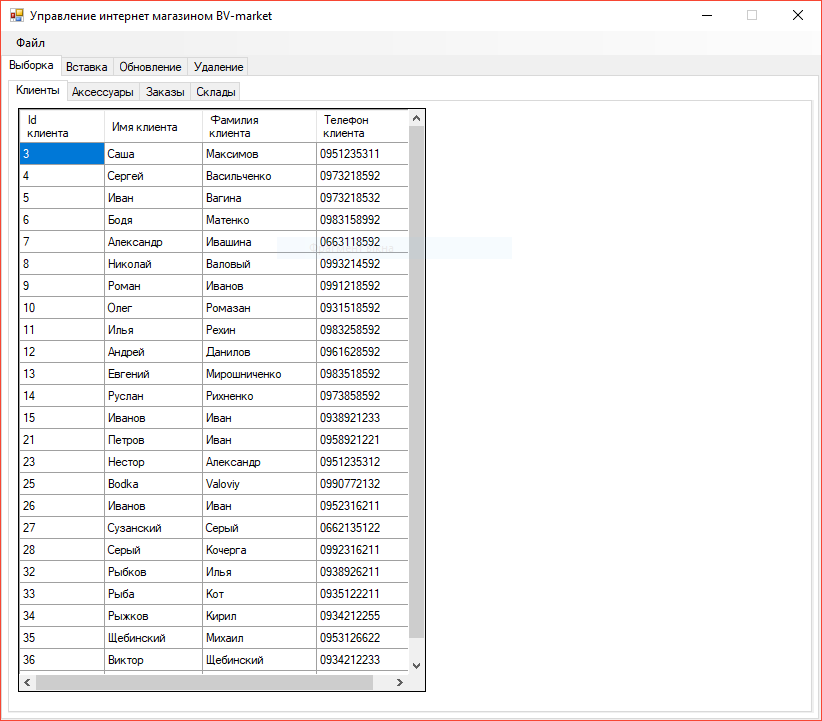


Рис. 1.4 – Форма выборки данных о клиенте из БД

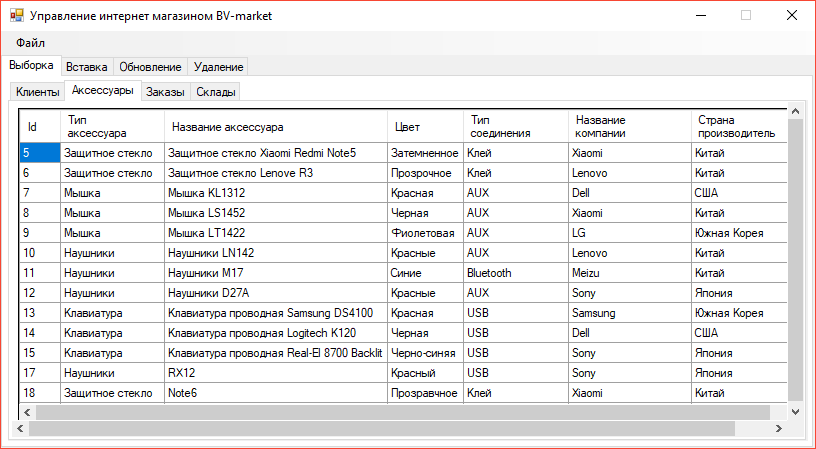


Рис 1.5 – Форма выборки данных о аксессуарах из БД

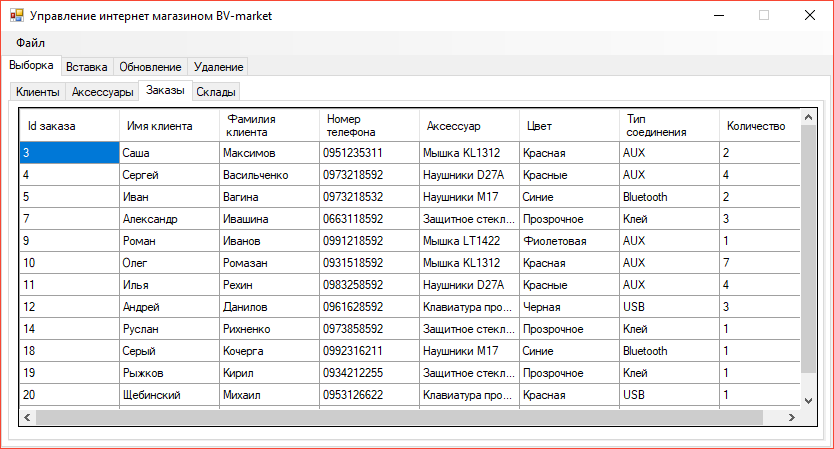


Рис 1.6 – Форма выборки данных о заказах из БД

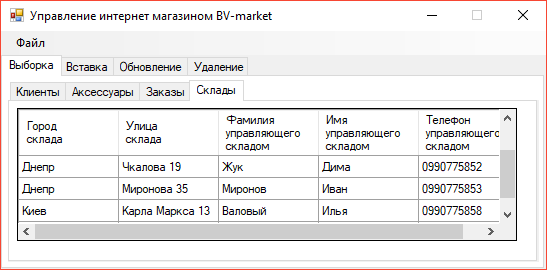


Рис 1.7 – Форма выборки данных о складах из БД

Формы вставки данных в таблицы. На рисунках 1.8, 1.9, 2.0 изображены формы вставки данных в таблиц:

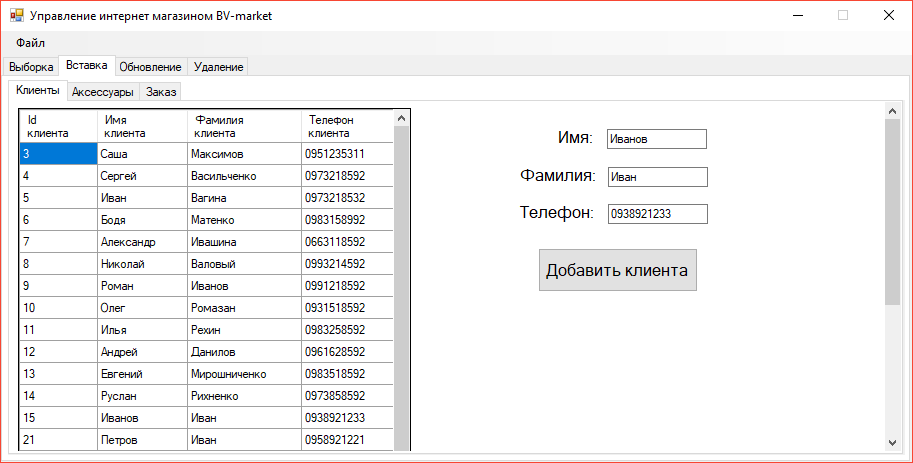


Рис 1.8 – Форма вставки данных о клиенте в БД

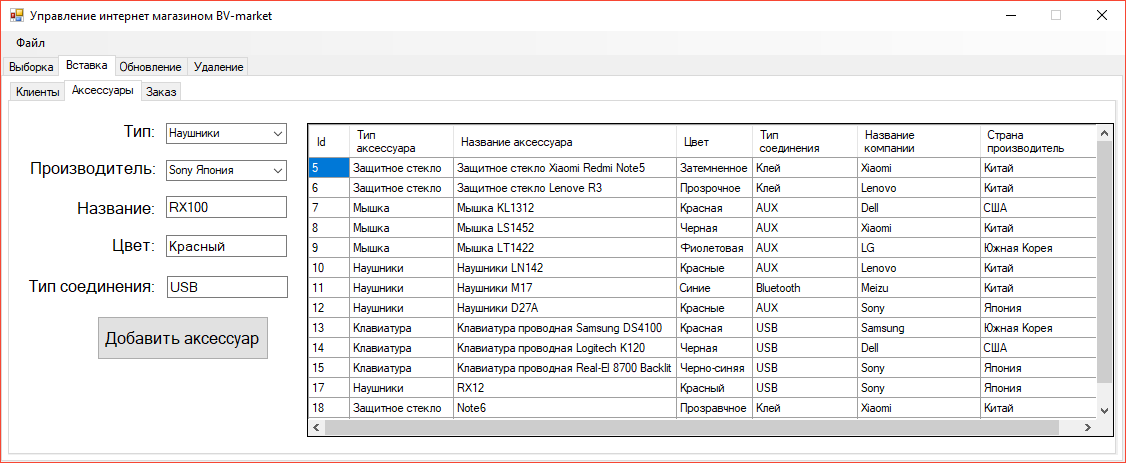


Рис 1.9 – Форма вставки данных о аксессуаре в БД

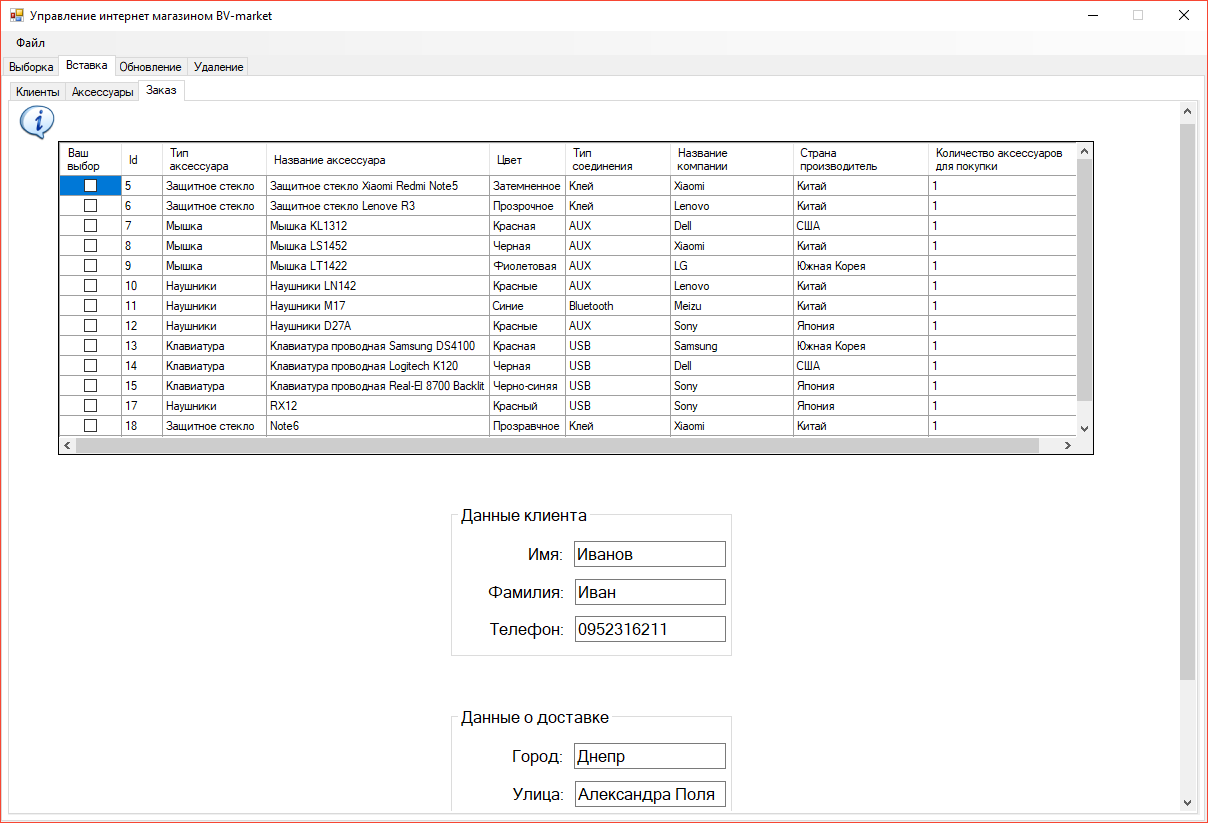


Рис 2.0 – Форма вставки данных о заказе в БД

Формы обновленя данных в таблицах. На рисунках 2.1, 2.2 изображены формы обновления данных в таблицах:

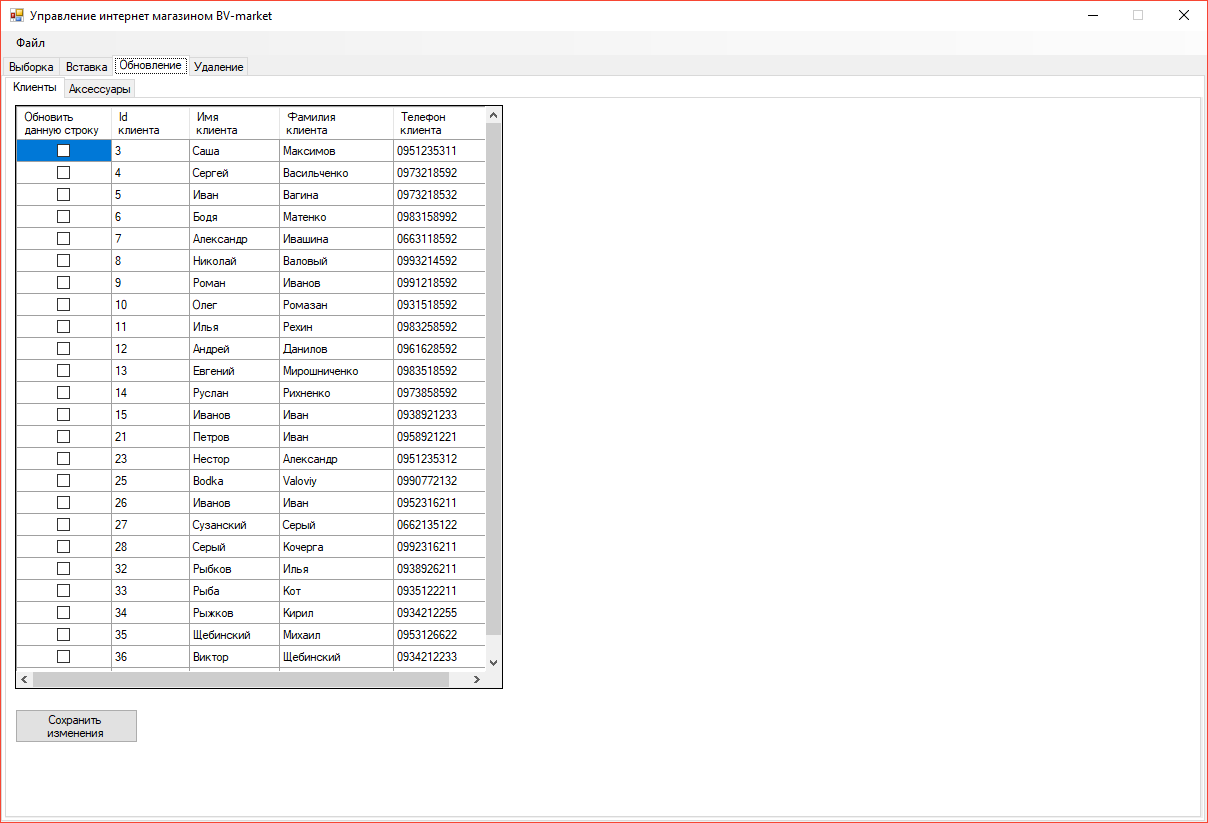


Рис 2.1 – Форма обновления данных о клиенте в БД

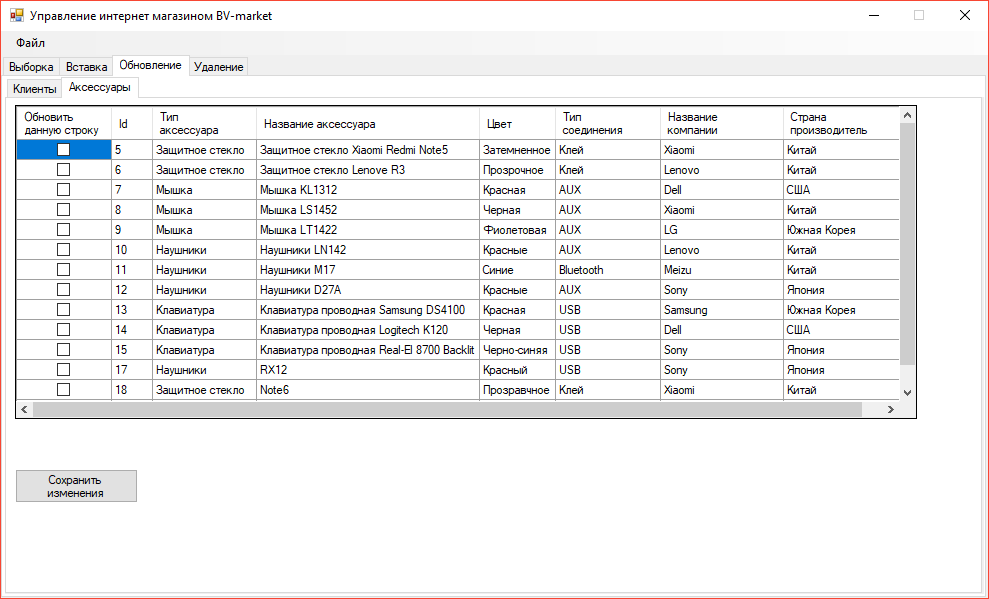


Рис 2.2 – Форма обновления данных о аксессуаре в БД

Формы удаления данных в таблицах. На рисунках 2.3, 2.4 изображены формы удаления данных в таблицах:

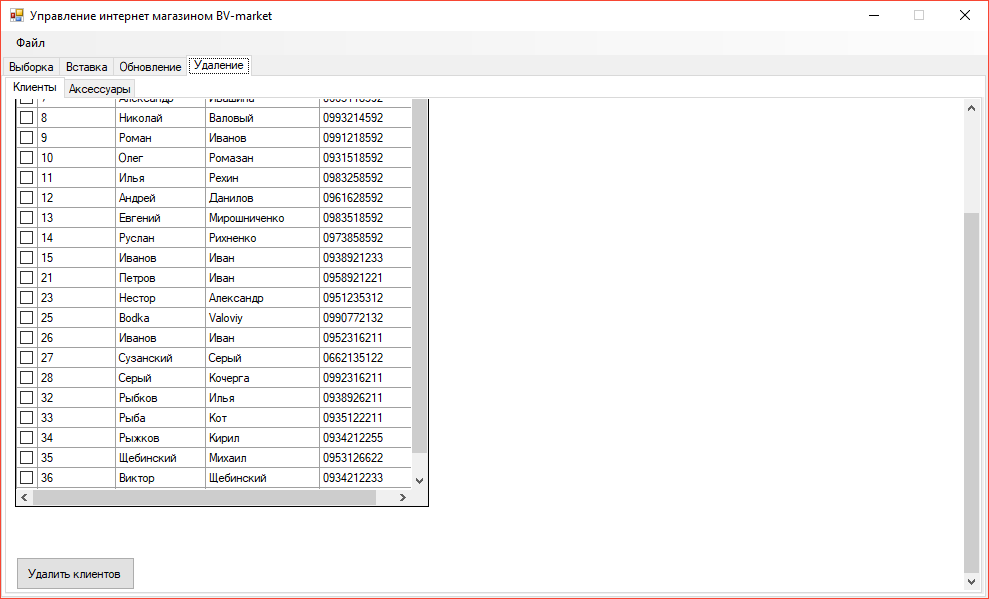


Рис 2.3 – Форма удаления данных о клиенте в БД

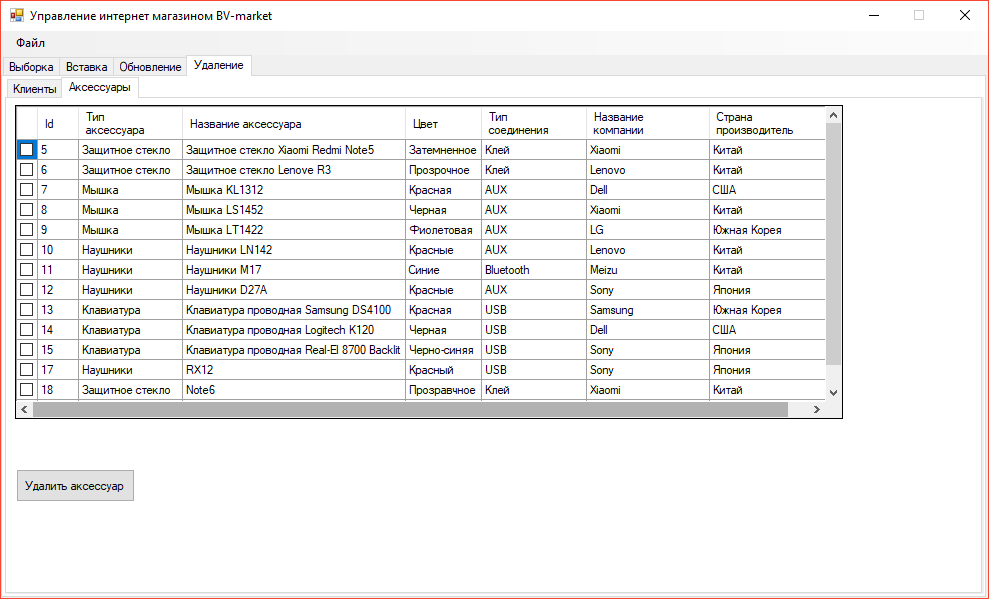


Рис 2.4 – Форма удаления данных о аксессуаре в БД

* 1. **Контрольные испытания по управлению БД, реализация запросов (соответствие картам транзакций****)**

Для проверки правильности работы разработанных запросов были проведены контрольные испытания по управлению БД. На рис. 2.5-3.1 изображены примеры выполнения некоторых запросов. На рисунках можно увидеть, что разработанные запросы соответствуют картам транзакций, которые описаны выше.

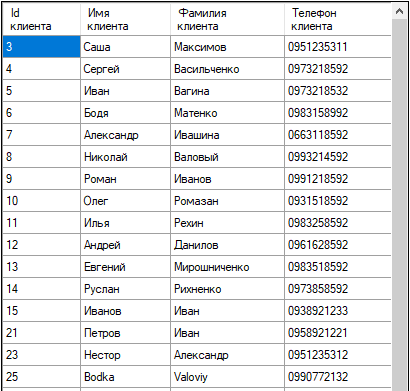


Рис. 2.5 – Выборка клиентов



Рис 2.6 – Выборка аксессуаров



Рис. 2.7 – Выборка заказов

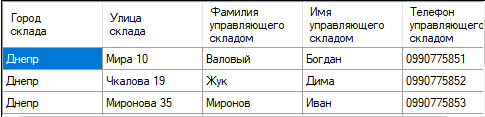


Рис. 2.8 – Выборка складов

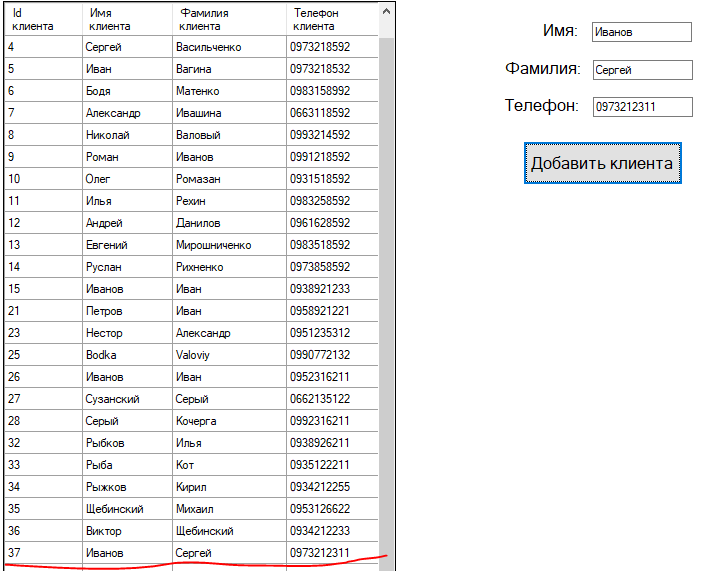


Рис 2.9 – Вставка нового клиента

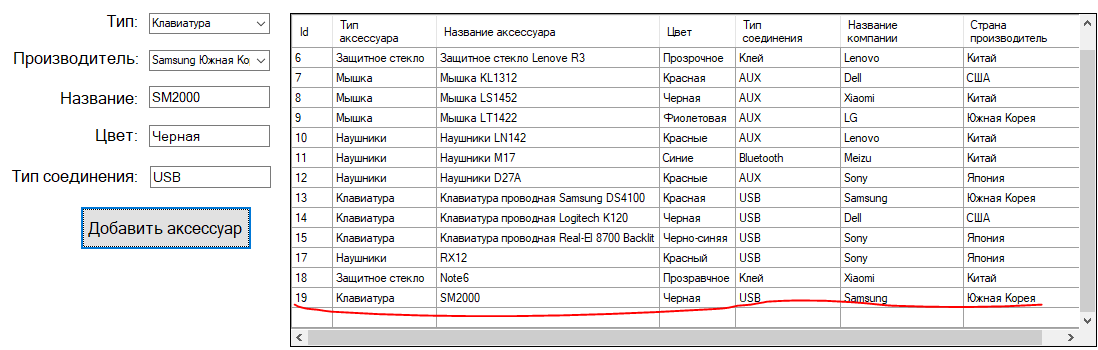


Рис 3.0 – Вставка нового аксуссара

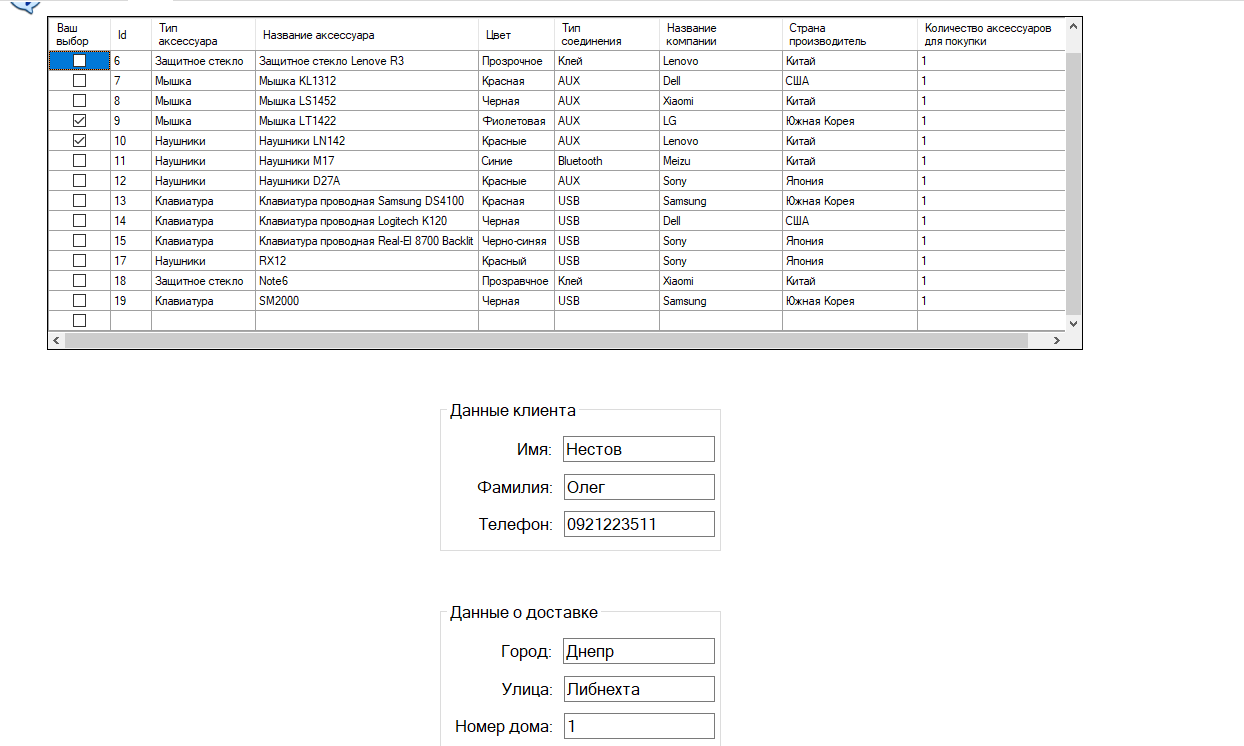




Рис. 3.1 – Вставка нового заказа

# 4 Проектирование реляционных аналитических систем накопления и обработки данных – ROLAP

ROLAP (реляционная OLAP) – OLAP-системы, которые имеют прямой доступ к существующим базам данных или используют данные, выгруженные в собственные локальные таблицы.

ROLAP предназначен для многомерного анализа данных, метаданных и вычисленных агрегатов. Настоящий ROLAP-продукт обеспечивает многомерный анализ данных, хранящихся в реляционной БД, и может работать с любой стандартной реляционной СУБД. Для физической реализации многомерной модели данных используется реляционный сервер БД.

Можно выделить следующие достоинства и недостатки ROLAP-систем:

1. Преимущества:

• ROLAP считается более масштабируемой для обработки больших объемов данных, особенно для моделей очень большой размерности (т.е. с миллионами элементов).

• Благодаря разнообразию доступных инструментов для загрузки данных, а также возможности для точной настройки ETL кода для конкретной модели данных, время загрузки данных, как правило, намного меньше, чем в MOLAP-системах.

• Данные хранятся в стандартной реляционной БД и доступ к ним можно получить, используя любой SQL-инструмент создания отчетности (кроме OLAP-инструментов).

• ROLAP-системы могут использовать элементы управления авторизацией БД, такие как безопасность на уровне строки, в результате чего результаты запроса фильтруются в зависимости от заданных критериев, устанавливаемых, например, для определенного пользователя или группы пользователей (оператор SQL WHERE).

2. Недостатки:

• Загрузка сводных таблиц должно управляться пользовательским ETL кодом. ROLAP-инструменты не помогают в решении этой задачи, что предполагает расходование больше времени на разработку и больше кода для поддержки данной функции.

• Если пропустить этап создания сводных таблиц, то производительность запросов страдать из-за обращения данных запросов к более детализированных таблиц. Это можно частично решить присоединением дополнительных сводных таблиц, однако по-прежнему является нецелесообразным создание таких таблиц для всех комбинаций размерности / атрибуты.

• Поскольку ROLAP-инструменты используют SQL для всех вычислений, то они не пригодны для моделей, где необходимо делать сложные вычисления, которые невозможно осуществить благодаря SQL. Примерами таких моделей является бюджетирование, финансовая отчетность и другие.

## 4.1 Выбор предметной области, исследование требований к хранилищу данных, составление предварительного набора аналитических запросов

Предметной областью для хранилища данных является модель продаж в интернет-магазине аксесуаров для ноутбуков и телефонов.

Сформулируем основные требования к хранилищу данных:

• Данные должны объединяться в категории и храниться в соответствии с областями, которые они описывают.

• Получение данных из хранилища должно происходить с высокой скоростью.

• Данные должны объединятся так, чтобы они удовлетворяли всем требованиям определенной области в целом, а не единственной ее функции.

• Данные в хранилище точны и корректны только в том случае, когда они привязаны к некоторому промежутку или моменту времени.

• Процесс пополнения данных должен быть качественно реализован.

• Хранилище должно иметь поддержку внутренней непротиворечивости данных.

Составим предварительный набор аналитических запросов к хранилищу данных:

• Запрос, отображающий информацию о всех продажах

• Запрос, который выводит информацию о продажах в определенных города

• Запрос, отображающий информацию о продажах определенного типа аксессуара

• Запрос, отображающий информацию о продажах определенного аксессуара

## 4.2 Разработка модели хранилища – выбор мер, измерений, таблиц фактов

При разработки модели хранилища в качестве таблицы фактов была выбрана таблица Facts. Она содержит следующие внешние ключи:

* ID заказа.
* ID клиента

Также были выбраны такие измерения:

* Заказы.
* Клиенты.
* Товары.
* Аксессуары.
* Производители.
* Тип аксессуаров.

Атрибуты измерения «Заказы»:

* ID (PK).
* Id клиента(FK).
* Город доставки.
* Улица доставки.
* Номер дома доставки.

Атрибуты измерения «Клиенты»:

* ID (PK).
* Имя.
* Фамилия.
* Номер.

Атрибуты измерения «Товары»:

* ID заказа (FK).
* ID аксессуара(FK).
* Количество.

Атрибуты измерения «Аксессуары»:

* ID (ПК).
* Имя.
* Цвет.
* Тип соединения.
* ID производителя(FK).
* ID типа(FK).

Атрибуты измерения «Производители»:

* ID (ПК).
* Название компании.
* Страна.

Атрибуты измерения «Тип аксессуаров»:

* ID (ПК).
* Название.

## 4.3 Графические модели нормализированных (снежинка) и ненормализованных (звезда) схем хранилища

На рис. 3.2 изображена нормализированная схема (снежинка) хранилища данных.

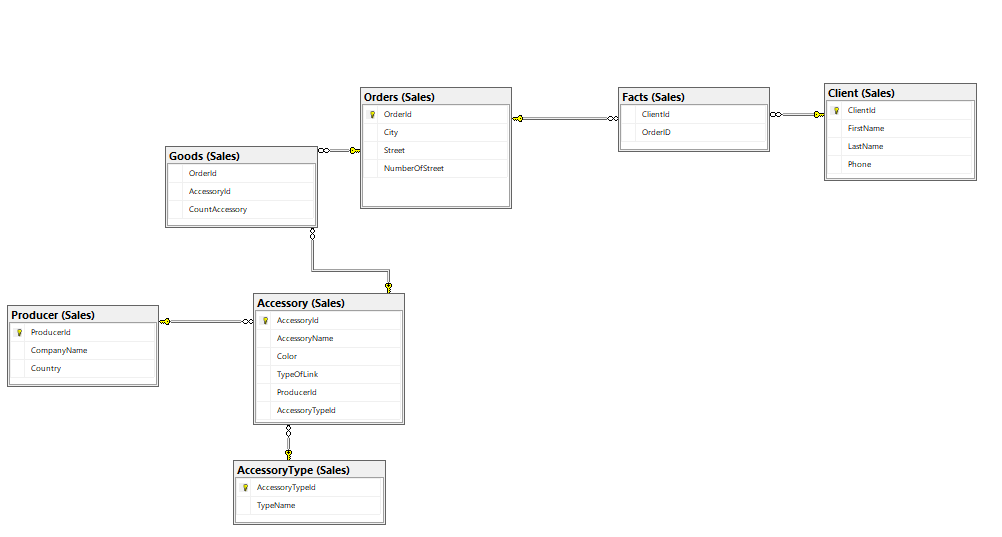


Рисунок 3.2 – Нормализированная схема (снежинка) ХД

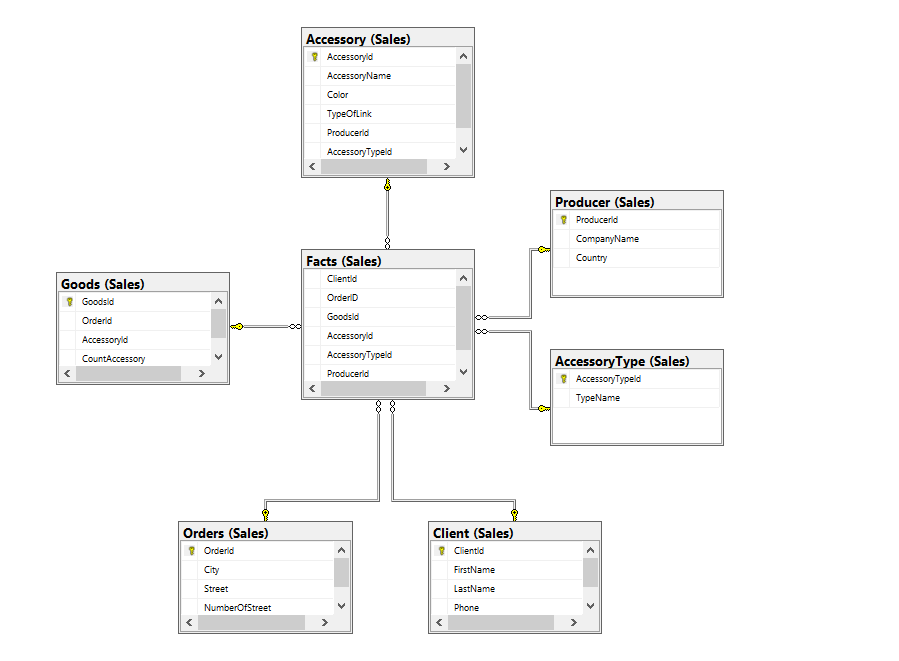


Рисунок 3.3 - не нормализированная схема (звезда) ХД.

## 4.4 Разработка физического проекта хранилища данных. Скрипт создания хранилища данных

/\* Создание Хранилища Данных\*/

-------------------------------------------------------------------------

CREATE DATABASE ShopDW

ON

(

NAME = 'ShopDW',

FILENAME = 'G:\Studying\4\_course\DataBase\CourseWorkDB\DW\ShopDW.mdf',

SIZE = 200MB,

MAXSIZE = 1000MB,

FILEGROWTH = 100MB

)

LOG ON

(

NAME = 'LogShopDW',

FILENAME = 'G:\Studying\4\_course\DataBase\CourseWorkDB\DW\LogShopDW.ldf',

SIZE = 100MB,

MAXSIZE = 500MB,

FILEGROWTH = 50MB

)

COLLATE Cyrillic\_General\_CI\_AS -- Задаем кодировку для базы данных по умолчанию

-------------------------------------------------------------------------

USE ShopDW;

GO

CREATE SCHEMA Sales

GO

--таблица клиентов

CREATE TABLE Sales.Client

(

ClientId int PRIMARY KEY IDENTITY(1,1) NOT NULL,

FirstName nvarchar(20) NOT NULL,

LastName nvarchar(20) NOT NULL,

Phone char(10) UNIQUE

)

-- Создание ограничения на телефон

ALTER TABLE Sales.Client

ADD CONSTRAINT CN\_ClientsPhone

-- Ограничение CHECK

CHECK (Phone LIKE '0[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

GO

--таблица заказов

CREATE TABLE Sales.Orders

(

OrderId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

City nvarchar(30) NOT NULL,

Street nvarchar(30) NOT NULL,

NumberOfStreet nvarchar(10) NOT NULL

)

GO

CREATE TABLE Sales.Facts

(

ClientId int NOT NULL

FOREIGN KEY REFERENCES Sales.Client(ClientId),

OrderID int NOT NULL

FOREIGN KEY REFERENCES Sales.Orders(OrderId)

)

GO

--производитель товаров

CREATE TABLE Sales.Producer

(

ProducerId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

CompanyName nvarchar(20),

Country nvarchar(20)

)

GO

--тип аксесуара

CREATE TABLE Sales.AccessoryType

(

AccessoryTypeId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

TypeName nvarchar(20) NOT NULL

)

GO

--аксесуар

CREATE TABLE Sales.Accessory

(

AccessoryId int IDENTITY(1,1) NOT NULL

PRIMARY KEY,

AccessoryName nvarchar(60) NOT NULL,

Color nvarchar(30) NOT NULL,

TypeOfLink nvarchar(30) NOT NULL,

ProducerId int NOT NULL

FOREIGN KEY REFERENCES Sales.Producer(ProducerId),

AccessoryTypeId int NOT NULL

FOREIGN KEY REFERENCES Sales.AccessoryType(AccessoryTypeId)

)

GO

--товар

CREATE TABLE Sales.Goods

(

OrderId int NOT NULL,

AccessoryId int NOT NULL

FOREIGN KEY REFERENCES Sales.Accessory(AccessoryId),

CountAccessory smallint NOT NULL

)

GO

ALTER TABLE Sales.Goods

ADD CONSTRAINT FK\_OrderId

FOREIGN KEY (OrderId) REFERENCES Sales.Orders(OrderId)

ON DELETE CASCADE ON UPDATE CASCADE

GO

--Наполнение хранилища данных

USE ShopDW

GO

-- добавляем производителей

INSERT INTO Sales.Producer

VALUES

('Sony', 'Япония'),

('Panasonic', 'Япония'),

('Dell', 'США'),

('Xiaomi', 'Китай'),

('Samsung', 'Южная Корея'),

('LG', 'Южная Корея'),

('Meizu', 'Китай'),

('Lenovo', 'Китай');

GO

-- добавляем типы аксессуаров

INSERT INTO Sales.AccessoryType

VALUES

('Наушники'),

('Мышка'),

('Защитное стекло'),

('Клавиатура'),

('Блютуз гарнитура');

GO

-- добавляем аксесуары

INSERT INTO Sales.Accessory

VALUES

('Bluetooth-гарнитура K213', 'Фиолетовая', 'Bluetooth', 8, 5),

('Bluetooth-гарнитура LE12', 'Синяя', 'Bluetooth', 2, 5),

('Bluetooth-гарнитура ST1243', 'Черная', 'Bluetooth', 1, 5),

('Защитное стекло Meizu M2', 'Прозрачное', 'Клей', 7, 3),

('Защитное стекло Xiaomi Redmi Note5', 'Затемненное', 'Клей', 4, 3),

('Защитное стекло Lenove R3', 'Прозрочное', 'Клей', 8, 3),

('Мышка KL1312', 'Красная', 'AUX', 3, 2),

('Мышка LS1452', 'Черная', 'AUX', 4, 2),

('Мышка LT1422', 'Фиолетовая', 'AUX', 6, 2),

('Наушники LN142', 'Красные', 'AUX', 8, 1),

('Наушники M17', 'Синие', 'Bluetooth', 7, 1),

('Наушники D27A', 'Красные', 'AUX', 1, 1),

('Клавиатура проводная Samsung DS4100', 'Красная', 'USB', 5, 4),

('Клавиатура проводная Logitech K120', 'Черная', 'USB', 3, 4),

('Клавиатура проводная Real-El 8700 Backlit', 'Черно-синяя', 'USB', 1, 4);

GO

-- добавляем клиентов

INSERT INTO Sales.Client

VALUES

('Максим', 'Петров', '0932345512'),

('Ваня', 'Васильев', '0953352112'),

('Саша', 'Максимов', '0951235311'),

('Сергей', 'Васильченко', '0973218592'),

('Иван', 'Тюрок', '0973218532'),

('Бодя', 'Матенко', '0983158992'),

('Александр', 'Ивашина', '0663118592'),

('Николай', 'Валовый', '0993214592'),

('Роман', 'Иванов', '0991218592'),

('Олег', 'Ромазан', '0931518592'),

('Илья', 'Рехин', '0983258592'),

('Андрей', 'Данилов', '0961628592'),

('Евгений', 'Мирошниченко', '0983518592'),

('Руслан', 'Рихненко', '0973858592');

GO

--добавляем заказы

INSERT INTO Sales.Orders

VALUES

('Днепр','Мира','17'),

('Днепр','Сокар','18'),

('Днепр','Щорса','19'),

('Днепр','Поля','23'),

('Киев','Карла Либнехта','64'),

('Киев','Винова','13'),

('Киев','Героев Майдана','76'),

('Киев','Алуева','43'),

('Киев','Ленина','21'),

('Киев','Немирова','85'),

('Киев','Липхи','12'),

('Харьков','Резолюции','62'),

('Харьков','Мудрого','27'),

('Харьков','Великого','93');

GO

-- добавляем товары

INSERT INTO Sales.Goods

VALUES

(1, 3, 3),

(2, 4, 1),

(3, 7, 2),

(4, 12, 4),

(5, 11, 2),

(6, 2, 6),

(7, 6, 3),

(8, 3, 2),

(9, 9, 1),

(10, 7, 7),

(11, 12, 4),

(12, 14, 3),

(13, 4, 6),

(14, 6, 1);

GO

--добавляем факты

INSERT INTO Sales.Facts

VALUES

(1,1),

(2,2),

(3,3),

(4,4),

(5,5),

(6,6),

(7,7),

(8,8),

(9,9),

(10,10),

(11,11),

(12,12),

(13,13)

GO

## 4.5 Реализация аналитических запросов к хранилищу данных средствами выбранной СУБД

USE ShopDW

GO

-- Запрос, отображающий информацию о всех продажах

SELECT

Sales.Orders.OrderId AS [Id заказа],

Sales.Orders.City AS [Город доставки],

Sales.Client.FirstName AS [Имя клиента],

Sales.Client.LastName AS [Фамилия клиента],

Sales.Client.Phone AS [Номер телефона],

Sales.Accessory.AccessoryName AS [Аксессуар],

Sales.Accessory.Color AS [Цвет],

Sales.Accessory.TypeOfLink AS [Тип соединения],

Sales.Goods.CountAccessory AS [Количество]

FROM Sales.Facts

INNER JOIN Sales.Orders

ON Sales.Orders.OrderId = Sales.Facts.OrderID

INNER JOIN Sales.Client

ON Sales.Client.ClientId = Sales.Facts.ClientId

INNER JOIN Sales.Goods

ON Sales.Orders.OrderId = Sales.Goods.OrderId

INNER JOIN Sales.Accessory

ON Sales.Accessory.AccessoryId = Sales.Goods.AccessoryId

GO

-- Запрос, который выводит информацию о продажах в определенных города

CREATE FUNCTION Sales.GetSales(@City nvarchar(20))

RETURNS TABLE

AS

RETURN SELECT

Sales.Orders.OrderId AS [Id заказа],

Sales.Client.FirstName AS [Имя клиента],

Sales.Client.LastName AS [Фамилия клиента],

Sales.Client.Phone AS [Номер телефона],

Sales.Accessory.AccessoryName AS [Аксессуар],

Sales.Accessory.Color AS [Цвет],

Sales.Accessory.TypeOfLink AS [Тип соединения],

Sales.Goods.CountAccessory AS [Количество]

FROM Sales.Facts

INNER JOIN Sales.Orders

ON Sales.Orders.OrderId = Sales.Facts.OrderID

INNER JOIN Sales.Client

ON Sales.Client.ClientId = Sales.Facts.ClientId

INNER JOIN Sales.Goods

ON Sales.Orders.OrderId = Sales.Goods.OrderId

INNER JOIN Sales.Accessory

ON Sales.Accessory.AccessoryId = Sales.Goods.AccessoryId

WHERE Sales.Orders.City = @City

GO

SELECT \* FROM Sales.GetSales('Днепр')

GO

--Запрос, отображающий информацию о продажах определенного типа аксессуара

CREATE FUNCTION Sales.GetSalesOfTypeAccessory(@TypeAccessory nvarchar(20))

RETURNS TABLE

AS

RETURN SELECT

Sales.Orders.OrderId AS [Id заказа],

Sales.Client.FirstName AS [Имя клиента],

Sales.Client.LastName AS [Фамилия клиента],

Sales.Client.Phone AS [Номер телефона],

Sales.Accessory.AccessoryName AS [Аксессуар],

Sales.Accessory.Color AS [Цвет],

Sales.Accessory.TypeOfLink AS [Тип соединения],

Sales.Goods.CountAccessory AS [Количество]

FROM Sales.Facts

INNER JOIN Sales.Orders

ON Sales.Orders.OrderId = Sales.Facts.OrderID

INNER JOIN Sales.Client

ON Sales.Client.ClientId = Sales.Facts.ClientId

INNER JOIN Sales.Goods

ON Sales.Orders.OrderId = Sales.Goods.OrderId

INNER JOIN Sales.Accessory

ON Sales.Accessory.AccessoryId = Sales.Goods.AccessoryId

INNER JOIN Sales.AccessoryType

ON Sales.AccessoryType.AccessoryTypeId = Sales.Accessory.AccessoryTypeId

WHERE Sales.AccessoryType.TypeName = @TypeAccessory

GO

SELECT \* FROM Sales.GetSalesOfTypeAccessory('Мышка');

GO

--Запрос, аксессуара отображающий информацию о продажах определенного аксессуара

CREATE FUNCTION Sales.GetSalesOfNameAccessory(@NameAccessory nvarchar(20))

RETURNS TABLE

AS

RETURN SELECT

Sales.Orders.OrderId AS [Id заказа],

Sales.Client.FirstName AS [Имя клиента],

Sales.Client.LastName AS [Фамилия клиента],

Sales.Client.Phone AS [Номер телефона],

Sales.Accessory.AccessoryName AS [Аксессуар],

Sales.Accessory.Color AS [Цвет],

Sales.Accessory.TypeOfLink AS [Тип соединения],

Sales.Goods.CountAccessory AS [Количество]

FROM Sales.Facts

INNER JOIN Sales.Orders

ON Sales.Orders.OrderId = Sales.Facts.OrderID

INNER JOIN Sales.Client

ON Sales.Client.ClientId = Sales.Facts.ClientId

INNER JOIN Sales.Goods

ON Sales.Orders.OrderId = Sales.Goods.OrderId

INNER JOIN Sales.Accessory

ON Sales.Accessory.AccessoryId = Sales.Goods.AccessoryId

WHERE Sales.Accessory.AccessoryName = @NameAccessory

GO

SELECT \* FROM Sales.GetSalesOfNameAccessory('Наушники M17');

GO

**4.6 Контрольные испытания аналитических запросов с использованием тестового набора данных**

Для проверки правильности работы запросов были проведены контрольные испытания с использованием тестового набора данных. На рисунках 3.3-3.7 изображены результаты выполнения нескольких запросов.

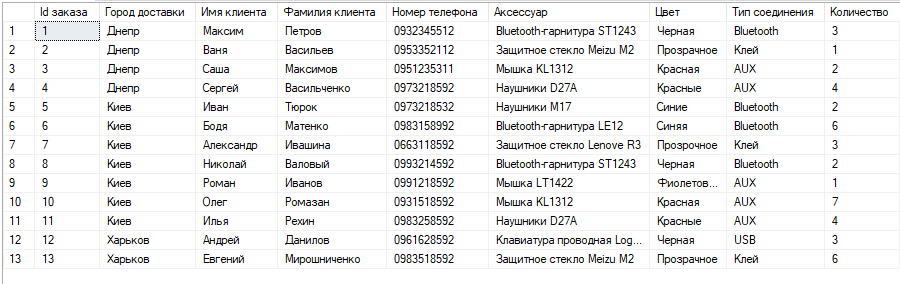


Рисунок 3.4 - информация о всех продажах

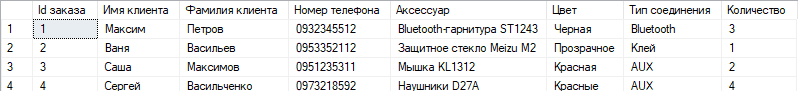


Рисунок 3.5 - информация о продажах в определенных города

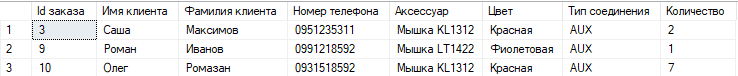


Рисунок 3.6 - информация о продажах определенного типа аксессуара



Рисунок 3.7 - информация о продажах определенного аксессуара

**Выводы**

В ходе выполнения курсового проекта были разработаны логические и физические проекты базы данных и хранилища данных. В результате проделанной работы были выяснены основные отличия баз данных от хранилищ данных.

В первую очередь хранилища данных содержат информацию, собранную из нескольких оперативных баз данных. Во-вторых, хранилища, как правило, на порядок больше баз данных, зачастую имея объем от сотен гигабайт до нескольких терабайт. В-третьих, хранилища данных создаются специально для приложений поддержки принятия решений и предоставляют накопленные за определенное время, сводные и консолидированные данные, которые более приемлемы для анализа, чем детальные индивидуальные записи.

Также можно выделить недостатки использования БД в принятии решений:

* недостоверность данных;
* низкая производительность при нестандартных запросах;
* невозможность преобразования разнородных данных, так как они часто не имеют меток времени;
* могут возникать проблемы при подготовке отчетов.

Отсюда следует, что обычные базы данных позволяют пользователю выполнять повседневную работу, а хранилища данных предназначены для принятия решений.

Подытожив все выше перечисленное, можно сделать такой вывод: если БД невелика, узкоспециализирована и имеется квалифицированный программист, способный писать нестандартные запросы, собирающие данные в массив и анализирующие эти данные, то вместо ХД можно использовать обычную БД. Хранилище же предназначено не для программиста - им должен суметь воспользоваться аналитик, руководитель, человек, не владеющий навыками написания сложных запросов.