# Содержание

[Введение 4](#_Toc248171444)

[1 Построение инфологической концептуальной модели 5](#_Toc248171445)

[1.1 Анализ предметной области и выявление необходимого набора сущностей 5](#_Toc248171446)

[1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов 7](#_Toc248171447)

[1.3 Определение связей между объектами 10](#_Toc248171448)

[1.4 Описание полученной модели на языке инфологического проектирования 12](#_Toc248171449)

[2 Построение схемы реляционной базы данных 13](#_Toc248171450)

[2.1 Построение набора необходимых отношений базы данных 13](#_Toc248171451)

[2.2 Задание первичных и внешних ключей определенных отношений 14](#_Toc248171452)

[2.3 Третья нормальная форма 15](#_Toc248171453)

[2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом 16](#_Toc248171454)

[2.5 Графическое представление связей между внешними и первичными ключами 17](#_Toc248171455)

[3 Создание спроектированной базы данных 18](#_Toc248171456)

[4 Запись выражений указанных в варианте задания типов запросов на языке SQL 25](#_Toc248171457)

[5 Выбор и основание средств разработки приложения 37](#_Toc248171458)

[6 Реализация законченного приложения, работающего с созданной базой данных 38](#_Toc248171459)

[6.1 Разработка и построение интерфейса главной и рабочих форм 38](#_Toc248171460)

[6.2 Построение главного меню и кнопок панели инструментов 39](#_Toc248171461)

[6.3 Выполнение программного кода на языке Java 39](#_Toc248171462)

[Заключение 41](#_Toc248171463)

[Список использованных источников 42](#_Toc248171464)

[Приложение A: Концептуальная схема БД 43](#_Toc248171465)

[Приложение B: Схема реляционной базы данных 44](#_Toc248171466)

[Приложение C: Главная и рабочие формы приложения 45](#_Toc248171467)

[Приложение D: Листинг программы 47](#_Toc248171468)

# Введение

Тема курсовой работы – проектирование реляционной базы данных магазин музыкальных инструментов.

В соответствии с предметной областью работы, будущая база данных должна хранить сведения о договорах, клиентах, инструментах, доставке, аренде, покупке и обеспечивать целостность этих данных.

Для обеспечения функционала, а также для удобства пользования информационной системой необходимо разработать приложение, которое позволит добавлять, удалять, редактировать и выводить информацию о клиентах, инструментах. Приложение должно быть простым в использовании, которым могли бы пользоваться даже неквалифицированные сотрудники.

Актуальность данной работы определяется тем, что в настоящее время в связи с развитием и распространением интернет-магазинов, то можно потом установить связь между интернет-магазином и базой данных, в которых и будет хранится вся информация. В таких целях возможно использование данного программного продукта. Он во много раз облегчит управление этим видом бизнеса.

Аналогов данной программы немало и не только в данной области. Они обычно разрабатываются под конкретную организацию, занимающуюся определенным видом деятельности. Создаваемая информационная система не привязывается к какому-либо существующему магазину, и создается на основе некого абстрактного магазина.

Для создания информационной базы данных будет использоваться СУДБ MsSQL. Для создания приложения – среда Visual Studio 2019.

# Построение инфологической концептуальной модели

## Анализ предметной области и выявление необходимого набора сущностей

В ходе анализа знаний и разработке базы данных были выявлены следующие основные сущности:

Сущность Договор описывает заказы, поступившие в магазин от клиентов, их типы и параметры. Характеризуется номером, сотрудником, который обслуживает клиента, клиентом, который осуществляет заказ, инструмент, который выбрал клиент, выбором между покупкой и арендой, а так же форма доставки инструмента.

Сущность Сотрудник описывает всех сотрудник, которые работают в магазине. Характеризуется фамилией, именем, отчеством, должностью, зарплатой и стажем работы.

Сущность Клиент описывает всех клиентов, сделавшие заказы или пользующиеся какими-либо услугами магазина. Характеризуется именем, фамилией, отчеством, адресом, паспортом, телефонным номером.

Сущность Инструменты описывает все инструменты магазина. Характеризуется названием, фирмой, производителем, личный номер, состоянием, в которым находится инструмент на данный момент и стоимость.

Сущность Состояние описывает состояние инструмента. Характеризуется датой проверки и дефектом.

Подсущность Состояния Дефект описывает дефекты инструментов. Характеризуется стоимостью за дефект.

Сущность Аренда характеризуется датой заключения, датой выдачи, датой возврата и стоимостью за аренду за определенный период.

Сущность Покупка характеризуется инструментом и его доставкой.

Сущность Доставка описывает выбранную клиентом доставку, характеризуется клиентом, датой доставки и формой доставки.

Подсущность Доставки *Форма доставки* описывает какая доставка предоставляется, характеризуется формой и стоимостью.

При начальном анализе также полезно строить диаграмму потоков данных (*англ.* Data Flow Diagram, DFD-diagram). Такая диаграмма описывает внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных для предметной области магазина представлена на рисунке 1.1.

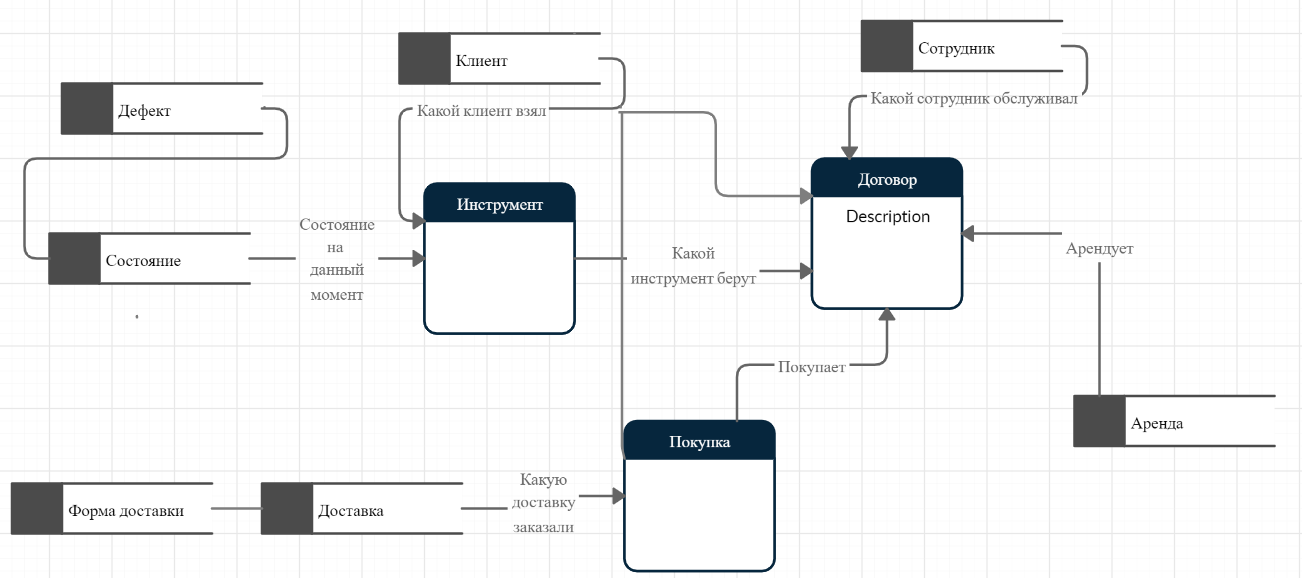


Рисунок 1.1 – Диаграмма потоков данных

По правилам построения DFD диаграмм [1] прямоугольником обозначаются источники и адресаты данных, прямоугольник с круглыми краями – функции, стрелками – потоки данных, на потоках обозначаются перемещаемые объекты.

## Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Ниже представлены сущности и определенные для них атрибуты, а также ключи (подчеркнуты). Имена сущностей и атрибутов указываются, как они будут определены в созданной базе данных, в скобках указывается перевод либо описание:

1. Contract (Договор):

* id (код контрактора);
* id\_client (код клиента);
* id\_employee (код сотрудника);
* id\_instrument (код инструмента);
* id\_rent (код аренды);
* id\_purchase (код покупки);
* numder (номер договора);

1. Client (Клиент):

* id;
* surname(фамилия);
* name(имя);
* patronymic(отчетсво);
* address(адрес);
* passport(паспорт);
* phone(телефон);

1. Defect (Дефект):

* id (код дефекта);
* name (название);
* cost (стоимость);

1. Delivery (Доставка):

* id (код доставки);
* data (дата доставки);
* id\_client (код клиента);
* id\_delivery\_form (код формы доставки);

1. Delivery\_Form (Форма доставки):

* id (код формы);
* form (форма доставки);
* cost (стоимость);

1. Employee (Сотрудник):

* id (код сотрудника);
* surname (фамилия);
* name (имя);
* patronymic (отчество);
* position (должность);
* salary (зарплата);
* work\_experince (стаж работы);

1. Instruments (Инструменты):

* id (код инструмента);
* name (название);
* firma (фирма);
* manufacturer (производитель);
* personal\_number (личный номер);
* id\_state (код состояния);
* cost(цена);

1. Purchase (Покупка):

* id (код покупки);
* id\_instruments (код инструмента);
* id\_delivery (код доставки);

1. Rent (Аренда):

* id (код аренды);
* conclusion (дата заключение);
* issue(дата выдачи);
* return (дата возврата);
* cost (стоимость);

1. State (Состояние):

* id (код состояния)
* check\_date (дата проверки);
* id\_defect (код дефекта);

## Определение связей между объектами

Кроме атрибутов каждой сущности модель данных должна определять связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между сущностями.

Связь – это ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако, одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся десятки или даже сотни сущностей, то между ними может быть установлено великое множество связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Для реализации информационной системы магазина необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему магазина в совокупности и определить взаимное влияние объектов, составляющих систему.

Этот процесс изображен на рис. 1.2

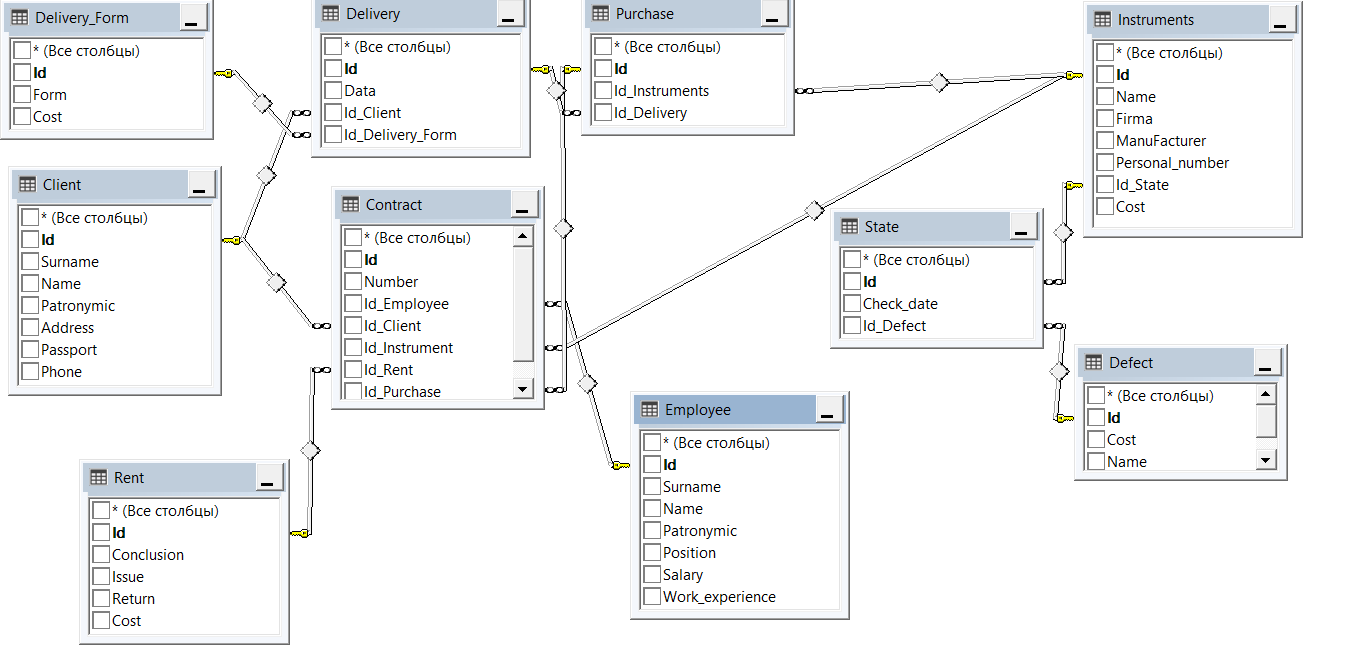


Рисунок 1.2 – Концептуальная схема базы данных

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рис. A.1 в приложении A.

## Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model). По правилам построения ER-диаграмм в нотации Питера Чена, сущности изображаются прямоугольниками, их атрибуты – овалами, отношения – ромбами. Связи между объектами изображаются линиями [2].

На основе проведенного проектирования, в частности на основе инфологической схемы, приведенной на рисунке 1.3, получим ER-диаграмму базы данных фотоцентра, представленную в Приложении A.

# Построение схемы реляционной базы данных

## Построение набора необходимых отношений базы данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте мы создали инфологическую концептуальную модель базы данных магазина, построенной с помощью языка «Таблицы-связи». На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных магазина.

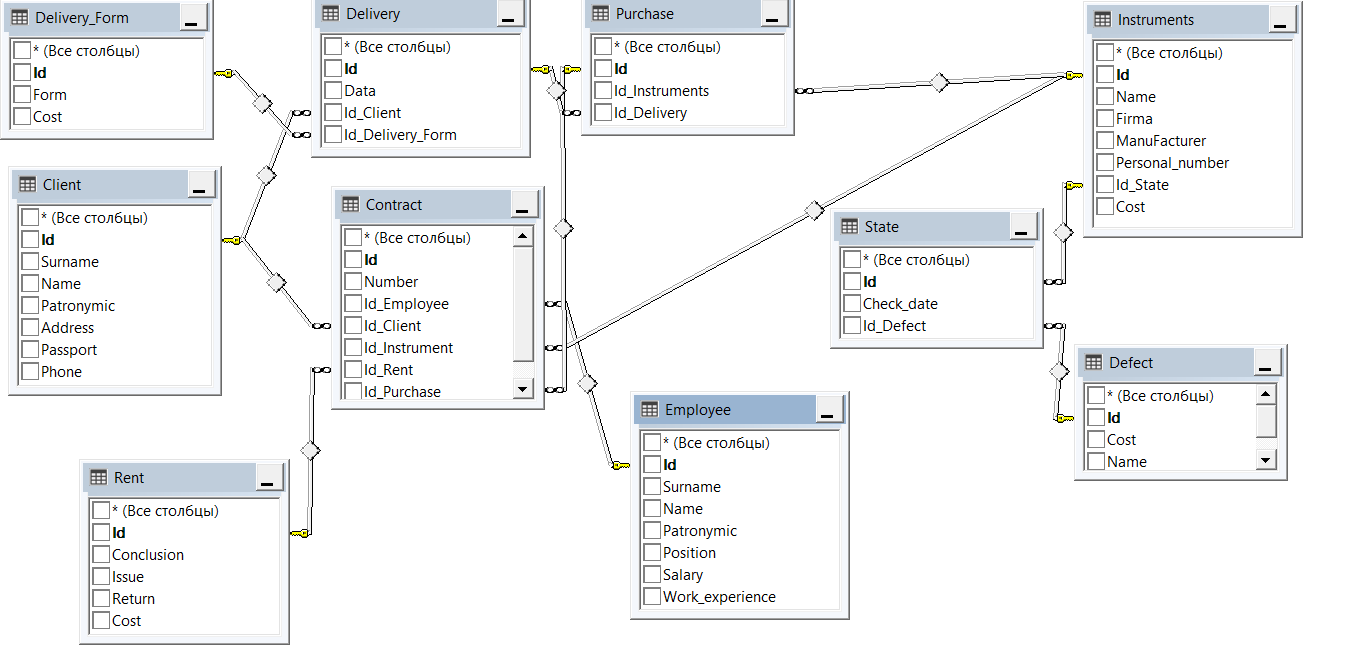


Рисунок 2.1 – Набор необходимых отношений базы данных

## Задание первичных и внешних ключей определенных отношений

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждое отношение должно обладать хотя бы одним ключом. В таблице 2.1 определены первичные и внешние ключи для отношений.

Таблица 2.1 – Первичные и внешние ключи отношений.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Первичный ключ | Внешние ключи |
|  | contract | id | id\_client (код клиента);  id\_employee (код сотрудника);  id\_instrument (код инструмента);  id\_rent (код аренды);  id\_purchase (код покупки); |
|  | client | id |  |
|  | defect | id |  |
|  | delivery | id | id\_client (код клиента);  id\_delivery\_form (код формы доставки); |
|  | delivery\_form | id |  |
|  | employee | id |  |
|  | instruments | id | id\_state (код состояния); |
|  | purchase | id | id\_instruments (код инструмента);  id\_delivery (код доставки); |
|  | rent | id |  |
|  | state | id | id\_defect (код дефекта); |

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

Таким образом, ненормализованная схема базы данных представлена в приложении B на рисунке B.1.

## Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым, исключив из базы данных избыточность и аномалии обновления.

Определение 3НФ – неключевые атрибуты не должны определять другие неключевые атрибуты.

Например, изначально Delivery была с полями Data, Id\_Client, Form\_Delivery, форма доставки может быть разной и по цене будет тоже разной, что вызывает проблему.

Поэтому необходимо привести эту таблицу к 3НФ. Результатом приведения будут две таблицы, приведенных на рисунке 2.3.

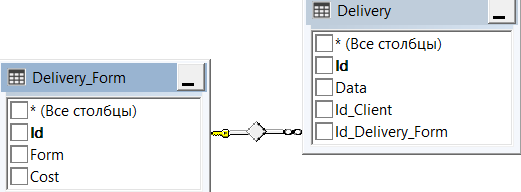


Рисунок 2.3 – Результат приведения таблицы Delivery к 3НФ

Процесс нормализации применяем также к таблице Instruments добавив туда таблицу State, в свою очередь к таблице состояния добавляется таблица Defect.

## Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

У проектируемой базы следующие таблицы будут иметь первичные ключи: *contract, client, defect,*

*Delivery, delivery\_form, employee, instruments, purchase, rent, state.*

Все связи имеют мощность 1-\* .

## Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении B на рисунке B.2. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются буквами PK (*от англ.* Primary Key – первичный ключ), внешние ключи обозначаются FK (*от англ.* Foreign Key – внешний ключ).

# Создание спроектированной базы данных

Для реализации спроектированной базы данных была выбрана система управления базами данных MsSQL. Это обусловлено тем, что, во-первых, MsSQL довольно широко распространена, во-вторых, является свободно распространяемой, и, в-третьих, MsSQL поддерживается большинством применяемых средств доступа к данным.

В создаваемой базе данных будут использоваться следующие типы данных [3]:

1. INT – Целочисленный тип. Размер – 4 байта
2. NCHAR – Строковый тип
3. DATE – Тип, определяющий дату.

Опишем все таблицы, которые будут созданы в базе данных.

Таблица contract содержит перечень всех сделанных заказов в целом по магазину музыкальных инструментов. Ее структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы *contract*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | INT | Идентификатор заказа. Ключевой атрибут |
| Number | INT | Личный номер договора |
| Id\_Employee | INT | Идентификатор сотрудника, который обслуживает. Необязательный атрибут |
| Id\_Client | INT | Идентификатор клиента, который делает заказ.  Необязательный атрибут |
| Id\_Instrument | INT | Идентификатор инструмента. Необязательный атрибут |
| Id\_Rent | INT | Идентификатор аренды инструмента. Необязательный атрибут |
| Id\_Purchase | INT | Идентификатор покупки инструмента. Необязательный атрибут |

Таблица delivery\_form содержит все существующие формы доставки, которые могут быть предложены клиентам. Ее структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы delivery\_form.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор формы доставки. Ключевой атрибут |
| form | NCHAR(50) | Название формы |
| cost | INT | Стоимость данной формы доставки |

Таблица *defect* содержит все существующие типы дефектов, которые могут быть у инструментов. Ее структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы *defect*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор дефекта инструмента. Ключевой атрибут |
| name | NCHAR(30) | Название дефекта |
| cost | INT | Стоимость каждого дефекта |

Таблица *delivery* содержит все существующие типы доставки, которые могут быть предложены клиентам. Ее структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы *delivery*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор доставки. Ключевой атрибут |
| Data | DATE | Дата доставки инструмента |
| Id\_Client | INT | Идентификатор клиента, который делает заказ.  Необязательный атрибут |
| Id\_Delivery\_Form | INT | Идентификатор формы доставки инструмента. Необязательный атрибут |

Таблица *purchase* содержит покупку инструмента. Ее структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы *purchase*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор покупки инструмента. Ключевой атрибут |
| Id\_Instruments | INT | Идентификатор инструмента. Необязательный атрибут |
| Id\_Delivery | INT | Идентификатор доставки. Необязательный атрибут |

Таблица *rent* содержит все существующие типы бумаги, на которых фотографии могут быть напечатаны. Ее структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы rent.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор аренды. Ключевой атрибут |
| Conclusion | DATE | Дата заключения аренды |
| Issue | DATE | Дата выдачи инструмента |
| Return | DATE | Дата возвраты инструмента |
| Cost | INT | Стоимость аренды за период |

Таблица *instruments* содержит перечень всех оказанных услуг в целом по фотоцентру. Ее структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы *instruments*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор инструмента. Ключевой атрибут |
| Name | NCHAR(50) | Название инструмента |
| Firma | NCHAR(50) | Название фирмы |
| ManuFacturer | NCHAR(50) | Название производителя инструмента |
| Personal\_number | INT | Персональный номер инструмента |
| Id\_State | INT | Идентификатор состояния инструмента. Необязательный атрибут |
| Cost | INT | Стоимость инструмента |

Таблица state содержит все существующие типы услуг, которые могут быть предложены клиентам. Ее структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы state.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор состояния инструмента. Ключевой атрибут |
| Check\_date | DATE | Дата проверки состояния инструмента |
| Id\_Defect | INT | Идентификатор дефектов инструмента. Необязательный атрибут |

Таблица clients содержит перечень всех клиентов фотоцентра. Ее структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы clients.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор клиента. Ключевой атрибут |
| Name | NCHAR(50) | Имя клиента |
| Surname | NCHAR(50) | Фамилия клиента |
| Patronymic | NCHAR(50) | Отчество клиента |
| address | NCHAR(60) | Домашний адрес |
| Passport | NCHAR(10) | Паспортные данные клиента |
| Phone | NCHAR(13) | Номер телефона клиента |

Таблица *employee* описывает дискаунты клиентов. Ее структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы *employee*.

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| id | INT | Идентификатор сотрудника. Ключевой атрибут |
| Surname | NCHAR(50) | Фамилия сотрудника |
| Name | NCHAR(50) | Имя сотрудника |
| Patronymic | NCHAR(50) | Отчество сотрудника |
| Position | NCHAR(30) | Должность сотрудника |
| Salary | INT | Зарплата сотрудника на данный момент |
| Work\_experience | INT | Стаж работы сотрудника на данный момент |

# Запись выражений указанных в варианте задания типов запросов на языке SQL

**Триггер для обеспечения целостности данных (откаты).**

CREATE TRIGGER [Otk]

ON [dbo].[State]

FOR INSERT

AS

BEGIN

SET NOCOUNT ON

declare @ins\_id\_Defect int

set @ins\_id\_Defect = (select Id\_Defect from inserted)

declare @dob int

set @dob = (select Defect.Id from Defect where Id = @ins\_id\_Defect)

if (@dob is null)

begin

ROLLBACK TRAN

print ('~~~Not Null~~~')

end

END

**Триггер с использованием функций для работы со строковыми переменными.**

create trigger UpLettersTrigger

on [dbo].[Client]

instead of insert

as

begin

declare @ins\_Id int

declare @ins\_S nvarchar(50)

declare @ins\_N nvarchar(50)

declare @ins\_P nvarchar(50)

declare @ins\_Address nvarchar(60)

declare @ins\_Passport nvarchar(10)

declare @ins\_Phone nvarchar(13)

set @ins\_Id = (select Id from inserted)

set @ins\_S = (select lower(Surname) from inserted)

set @ins\_N = (select lower(Name) from inserted)

set @ins\_P = (select lower(Patronymic) from inserted)

set @ins\_Address = (select Address from inserted)

set @ins\_Passport = (select Passport from inserted)

set @ins\_Phone = (select Phone from inserted)

declare @new\_S nvarchar(50)

set @new\_S = upper(substring(@ins\_S,1,1)) + substring(@ins\_S,2,LEN(@ins\_S))

declare @new\_N nvarchar(50)

declare @new\_P nvarchar(50)

if (@ins\_N is not null)

begin

set @new\_N = upper(substring(@ins\_N,1,1)) + substring(@ins\_N,2,LEN(@ins\_N))

end

else

begin

set @new\_N = @ins\_N

end

if (@ins\_P is not null)

begin set @new\_P = upper(substring(@ins\_P,1,1)) + substring(@ins\_P,2,LEN(@ins\_P))

end

else

begin

set @new\_P = @ins\_P

end

Insert into Client values (@ins\_Id,@new\_S,@new\_N,@new\_P,@ins\_Address,@ins\_Passport,@ins\_Phone)

end

**Триггер с использованием условной конструкции IF.**

CREATE TRIGGER [Date]

ON [dbo].[Delivery]

instead of insert

as

begin

declare @ins\_id int

set @ins\_id = (select Id from inserted)

declare @ins\_Data date

set @ins\_Data = (select Data from inserted)

declare @ins\_id\_Client int

set @ins\_id\_Client = (select Id\_Client from inserted)

declare @ins\_id\_Delivery\_Form int

set @ins\_id\_Delivery\_Form = (select Id\_Delivery\_Form from inserted)

declare @check\_ins int -- 0 нормально, 1 ошибка

set @check\_ins = 0

if (

@ins\_id\_Client > max(Id\_Client)

)

begin

print ('Client does not exist ')

set @check\_ins = 1

end

if (

@ins\_id\_Delivery\_Form > max(Id\_Delivery\_Form)

)

begin

print ('Delivery does not exist ')

set @check\_ins = 1

end

if ( @check\_ins = 0 )

begin

insert into Delivery

values (@ins\_id,@ins\_Data,@ins\_id\_Client,@ins\_id\_Delivery\_Form)

print('Good')

end

end

**Триггер с использованием цикла WHILE.**

CREATE TRIGGER [FreeDelivery]

ON [dbo].[Delivery\_Form]

instead of update

as

begin

DECLARE @x INT, @x\_old INT, @y INT, @y\_old INT

-- курсор с новыми значениями

DECLARE CUR1 CURSOR FOR

SELECT Form,Cost

FROM inserted

-- курсор со старыми значениями

DECLARE CUR2 CURSOR FOR

SELECT Form,Cost

FROM deleted

OPEN CUR1

OPEN CUR2

-- перемещаемся параллельно по обоим курсорам

FETCH NEXT FROM CUR1 INTO @x, @y

FETCH NEXT FROM CUR2 INTO @x\_old, @y\_old

WHILE @@FETCH\_STATUS=0

BEGIN

--для старого кода уменьшается егo количество

UPDATE Delivery\_Form

SET Cost=Cost-@y\_old

WHERE Form=@x\_old

--для нового кода, если такого--еще нет на складе, вводится новая запись

IF NOT EXISTS (SELECT \* FROM Delivery\_Form

WHERE Form=@x)

INSERT INTO Delivery\_Form(Id,Form,Cost)

VALUES (@x,@y)

ELSE

--иначе для нового кода товара увеличивается

--его количество на складе

UPDATE Delivery\_Form

SET Cost=Cost+@y

WHERE Form=@x

FETCH NEXT FROM CUR1 INTO @x, @y

FETCH NEXT FROM CUR2 INTO @x\_old, @y\_old

END

CLOSE CUR1

CLOSE CUR2

DEALLOCATE CUR1

DEALLOCATE CUR2

end

**Вывод информации с таблицы**

create function GetClient()

returns table

as

return

(select Id, Surname, Name, Patronymic, Address, Passport, Phone

from Client

)

**Вывод ФИО в один столбец**

CREATE FUNCTION GetClienrt2()

returns table

with schemabinding

as

return

(select Id, Surname+''+Name+''+Patronymic as FIO

from dbo.Client

)

**Вывод минимальной цены инструмента**

create function MinCost()

returns table

as

return

(select MIN(Cost)

from Instruments

)

# Выбор и основание средств разработки приложения

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое). Другими показателями, влияющими на выбор СУБД, являются [2]:

* + удобство и простота использования;
  + качество средств разработки, защиты и контроля базы данных;
  + уровень коммуникационных средств в случае применения ее в сетях;
  + фирма-разработчик;
  + стоимость.

Система MsSQL Server позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. MsSQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MsSQL Server позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2019, в качестве языка программирования – C#.

Достоинства платформы .NET [1]:

1) Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику объединены в единую иерархию классов. Другими словами, все, что может вам потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем, все это сгруппировано очень удобно и интуитивно понятно.

2) Приложение, написанное на любом .NET-совместимом языке, является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3) В состав платформы .NET входит "сборщик мусора", который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

6) Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

7) .NET приложения могут быть сертифицированы на безопасность. Это является особенность промежуточного кода, в который преобразуются все .NET приложения.

8) Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

9) Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

# Реализация законченного приложения, работающего с созданной базой данных

## Разработка и построение интерфейса главной и рабочих форм

Прежде всего, в виду необходимости защиты информации, находящейся в базе, только после того, как пройдет удачная Windows-аутентификации пользователь может увидеть главную форму.

Главная форма построена на подобие MDI-форм: представляет собой окна, расположенные под одним общим окном. Главная форма содержит меню, разделенное на категории (подробнее в пункте 6.2). Все вторичные окна, за исключением диалоговых, являются вложенными - их нельзя вытащить за пределы формы, но можно развернуть на всю основную форму либо свернуть (при этом на нижней панели появится кнопка с соответствующим заголовком), что позволяет легко ориентироваться между окошками, открывать множество окон и предотвращает их «потерю» неопытным пользователем.

Внешний вид главной формы можно увидеть в приложении B (рис. B.1).

Основные формы приложения построены в основном на компоненте DataGridView. Данный компонент позволяет представлять данные в удобной табличной форме с использованием дополнительных собственных расширений.

При разработке интерфейса форм главным был принцип предотвращения ошибок, а не их констатации, потому были использованы (по возможности), к примеру, интерактивные обращения к базе еще на этапе заполнения форм. Таким образом, частично проверяется уникальность имен, номеров; также для всех полей, в которые должны быть занесены сугубо числовые данные, сделан соответствующий фильтр, что также позволяет предотвратить некорректные данные в базе и пугающие неопытного пользователя сообщения об ошибках.

## Построение главного меню и кнопок панели инструментов

Главное меню программы представлено шестью подменю: Список инструментов, база клиентов, фильтр, запрос, admin, бд. Список инструментов - это окно, где можно добавлять, редактировать и удалять инструменты. База клиентов - это окно, где можно добавлять, редактировать, удалять клиентов. Фильтр - это окно, в котором можно фильтровать записи о инструменте. В дальнейшем будет расширяться. Запрос - окно, в котором есть запрос на цену и дату взятия и возврата. Admin - это окно, в котором можно добавлять нового пользователя, а потом давать ему привелигии. БД - это окно, в котором можно быстро найти информацию о базе данных(таблицы, поля, пользователи).

## Выполнение программного кода на языке Java

Опишем работу приложения с базой данных. Для подключения к базе будем использовать SqlClient. Для подключения к базе данных использую данную строку:

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Кристина\source\repos\krbd\krbd\Database1.mdf;Integrated Security=True";

Для заполнения базы данных используется функция:

private void FillClient()

{

string SqlText = "SELECT \* FROM [Instruments]";

SqlDataAdapter da = new SqlDataAdapter(SqlText, ConnStr);

DataSet ds = new DataSet();

da.Fill(ds, "[Instruments]");

dataGridView1.DataSource = ds.Tables["[Instruments]"].DefaultView;

}

public void MyExecuteNonQuery(string SqlText)

{

SqlConnection cn; // экземпляр класса типа SqlConnection

SqlCommand cmd;

// выделение памяти с инициализацией строки соединения с базой данных

cn = new SqlConnection(ConnStr);

cn.Open(); // открыть источник данных

cmd = cn.CreateCommand(); // задать SQL-команду

cmd.CommandText = SqlText; // задать командную строку

cmd.ExecuteNonQuery(); // выполнить SQL-команду

cn.Close(); // закрыть источник данных

}

private void button1\_Click(object sender, EventArgs e)

{

string SqlText = "INSERT INTO [Instrument] ([ID], [Name], [Firma], [Manufacturer], [Personal\_number], [Id\_State], [Cost])";

Form2 f = new Form2(); // создать экземпляр окна

if (f.ShowDialog() == DialogResult.OK)

{

// сформировать SQL-строку

SqlText = "INSERT INTO [Instruments] ([ID], [Name], [Firma], [Manufacturer], [Personal\_number], [Id\_State], [Cost]) VALUES (";

SqlText = SqlText + "\'" + f.maskedTextBox1.Text + "\', ";

SqlText = SqlText + "\'" + f.textBox1.Text + "\', ";

SqlText = SqlText + "\'" + f.textBox2.Text + "\', ";

SqlText = SqlText + "\'" + f.textBox3.Text + "\', ";

SqlText = SqlText + "\'" + f.maskedTextBox2.Text + "\', ";

SqlText = SqlText + "\'" + f.comboBox1.Text + "\', ";

SqlText = SqlText + "\'" + f.maskedTextBox3.Text + "\')";

// выполнить SQL-команду

MyExecuteNonQuery(SqlText);

// отобразить таблицу Source

FillClient();

}

}

# Заключение

В результате выполненной курсовой работы была создана база данных фотоцентра, а также программный продукт, позволяющий пользователю взаимодействовать с базой данных. Базы данных была разработана в среде MySQL, приложение – Visual Studio 2019.

Приложение позволяет:

* просматривать различные данные по фотоцентру;
* вычислять некоторые параметры, такие как сумму выручки с заказов, с продаж и др.;
* добавлять и редактировать заказы, услуги и клиентов фотоцентра.

Запросы, указанные в варианте задания, были «растворены» в приложении.

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на Visual Studio 2019, проектировании баз данных и реализации их в СУБД MsSQL.

# Список использованных источников

1. Data flow diagram - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Data\_flow\_diagram
2. Entity-relationship model – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Entity-relationship\_model
3. MySQL 5.0 Reference Manual [Электронный ресурс] / Официальный сайт MySQL. Документация по MySQL. Режим доступа:

http://dev.mysql.com/doc/refman/5.0/en/index.html

1. Документация по языку Java от компании Sun [Электронный ресурс]. – Элек-трон. текстовые дан. (270 Мб). – Sun Microsystems, Inc., 2006
2. Документация по MySQL Connector/J [Электронный ресурс]. – Элек-трон. текстовые дан. (3 Мб). – Sun Microsystems, Inc., 2008
3. Unified Modeling Language – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Unified\_Modeling\_Language

Приложение A: Концептуальная схема БД

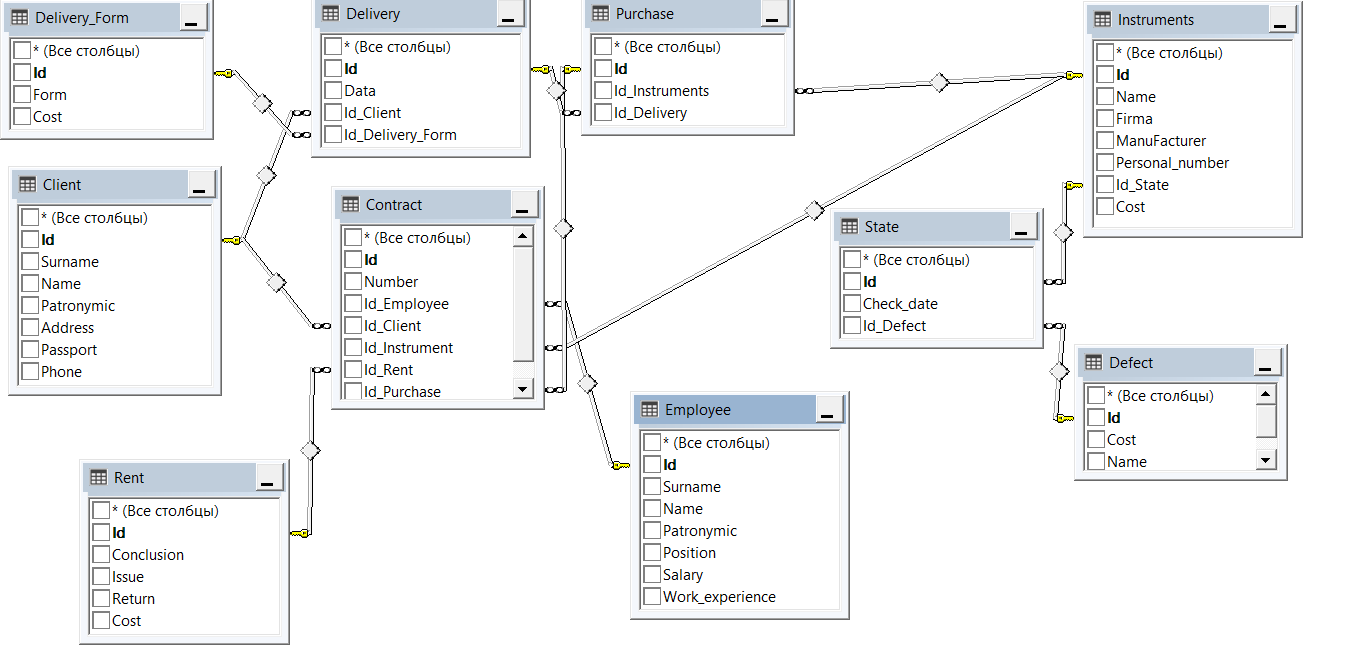


Рисунок A.1 – Инфологическая схема проектируемой базы данных

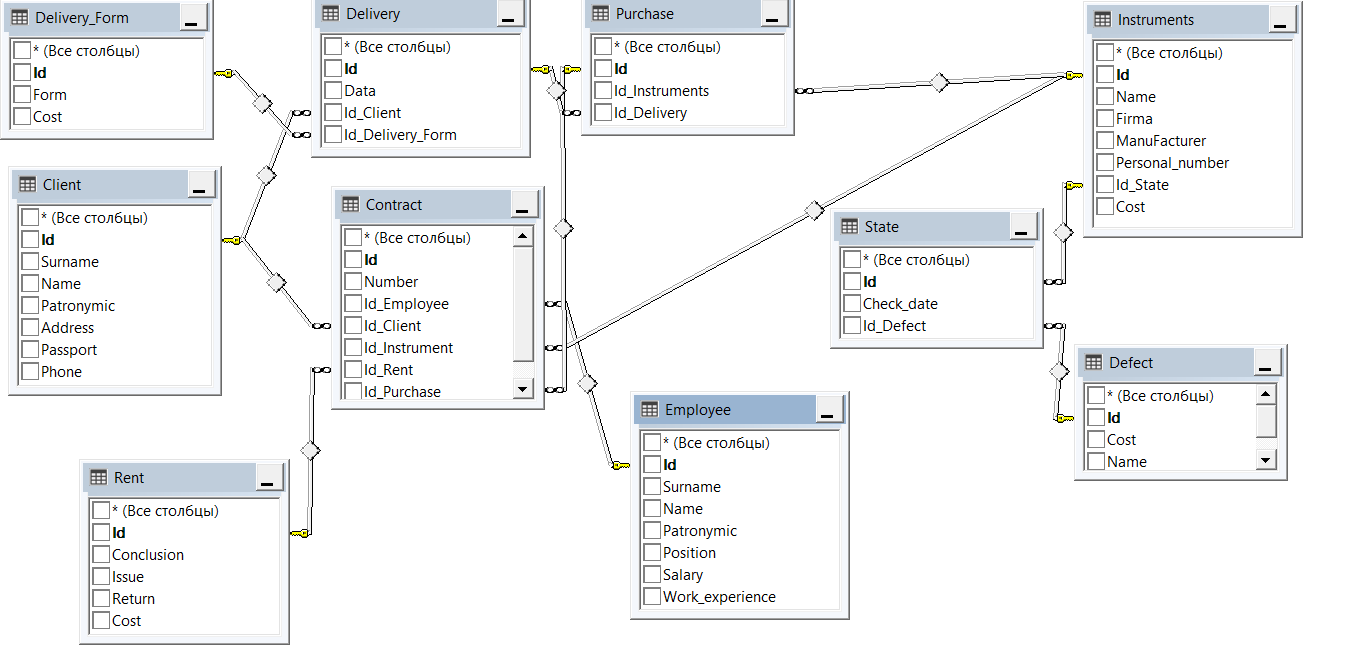


Рисунок A.2 – ER-диаграмма проектируемой базы данных

Приложение B: Схема реляционной базы данных

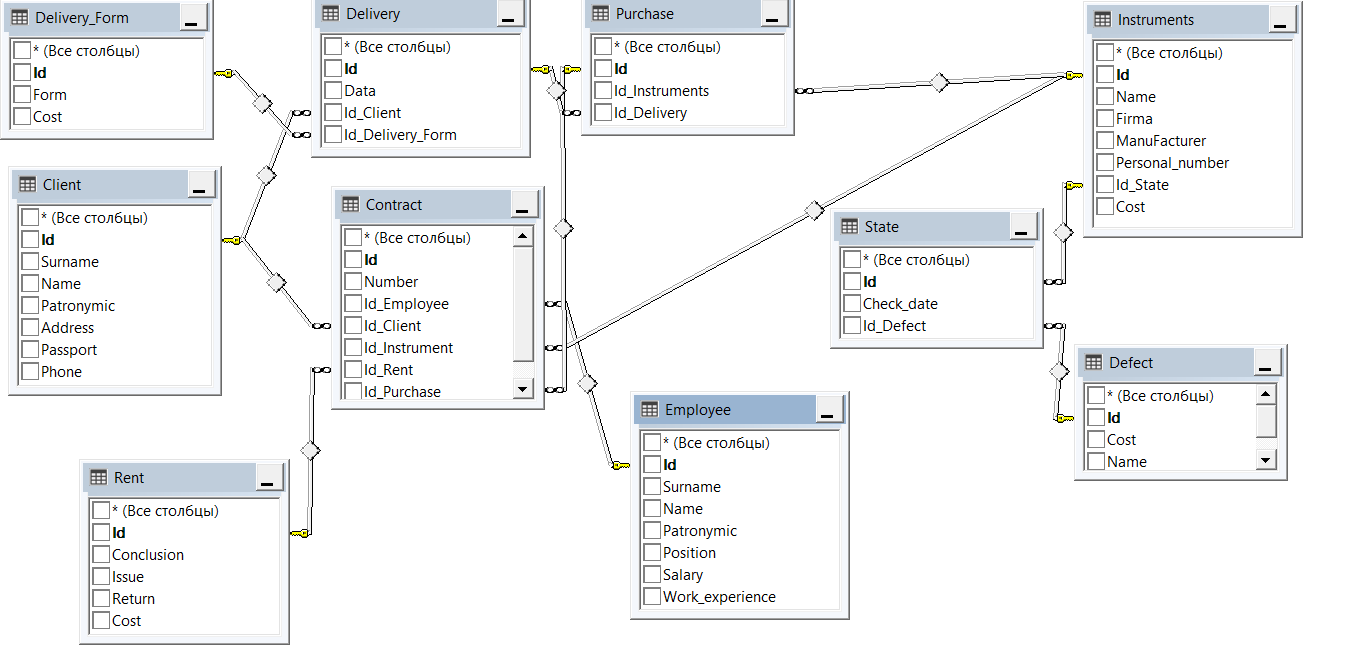


Рисунок B.2 – Схема после нормализации

Приложение C: Главная и рабочие формы приложения

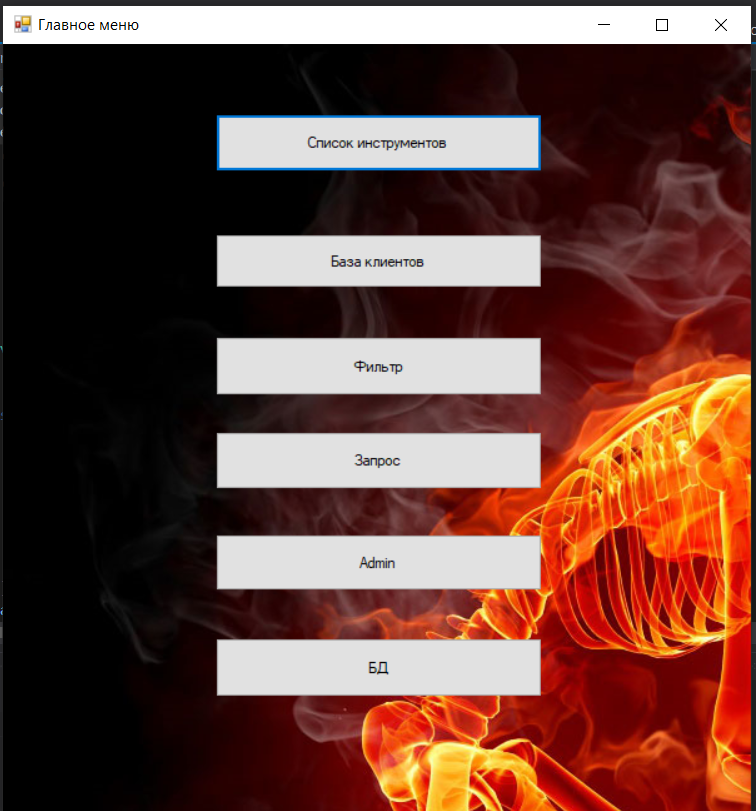


Рисунок C.1 – Главная форма приложения

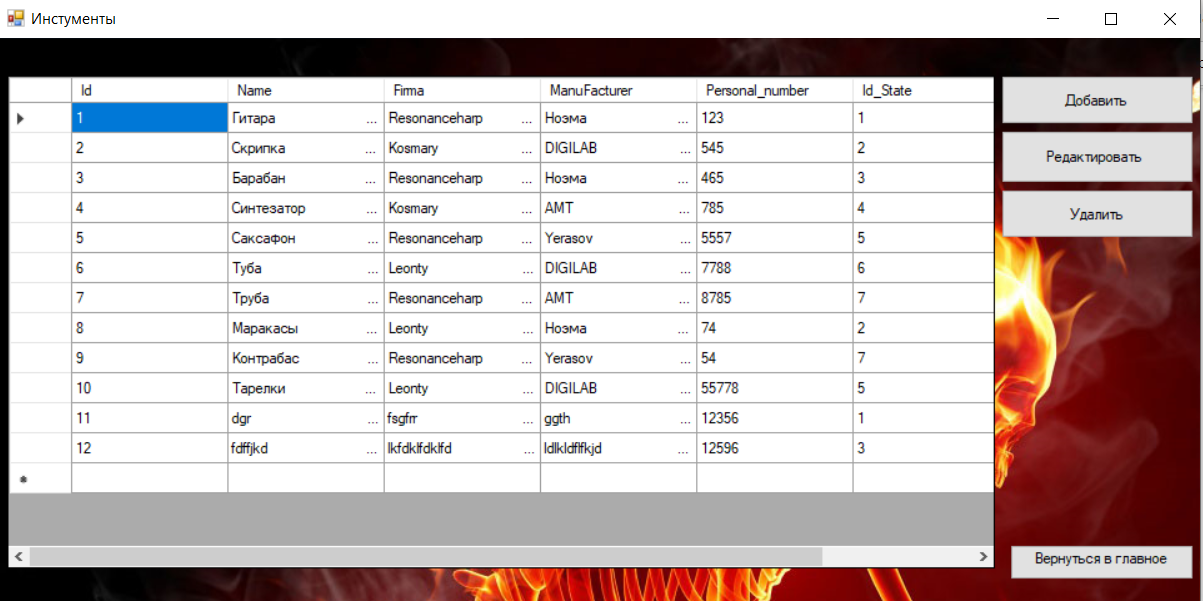
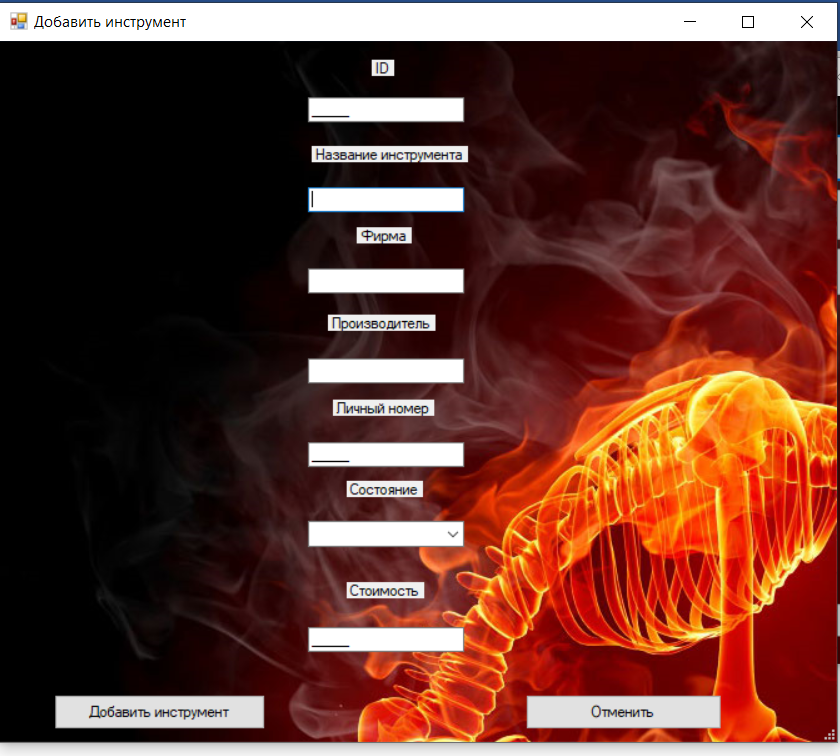
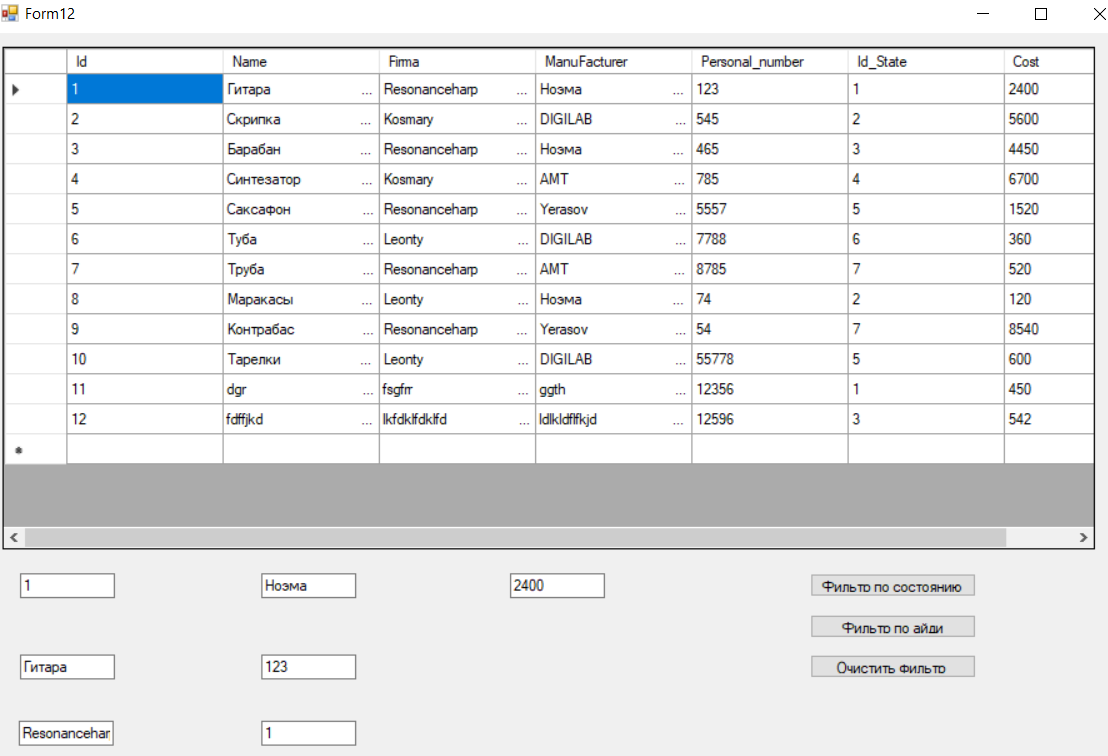


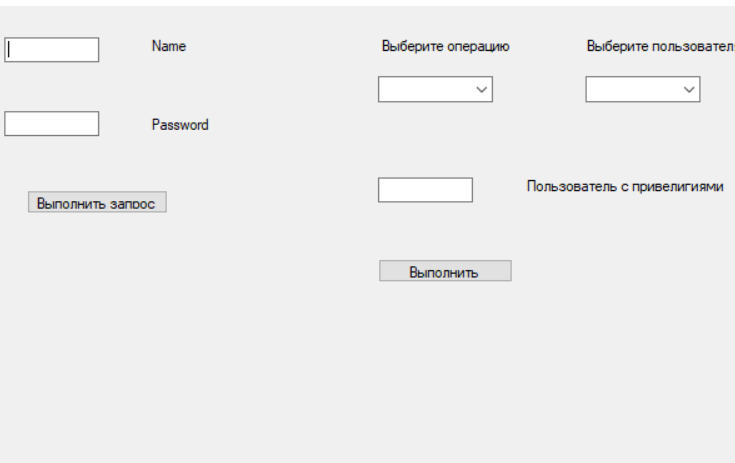
Рисунок C.2 – Форма приложения Списка инструментов



**Рисунок C.3 – Форма добавления инструмента**



**Рисунок С.4 - Форма фильтра**



**Рисунок С.5 - Форма администрирования**

Приложение D: Листинг программы