государственное автономное профессиональное образовательное учреждение   
Чувашской Республики «Межрегиональный центр компетенций –  
 Чебоксарский электромеханический колледж» Министерства образования и    
молодежной политики Чувашской Республики

Дисциплина МДК.11.01 Технология разработки и защита базы данных

КУРСОВОЙ ПРОЕКТ

Технология разработки и защиты базы данных

автоматизированной информационной системы складского учета

КП.Ип1-21.06.МДК.11.01.ПЗ

Выполнил студент 3 курса, группы Ип1-21

Иваков Алексей Петрович

(Фамилия И. О.)

(подпись) (чч.мм.гггг)

Преподаватель Игнатьева Т.А.

(Фамилия И. О.)

Защищен

(чч.мм.гггг)

с оценкой

Подпись

(подпись) (расшифровка подписи)

2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc121320230)

[1 Инфологическое проектирование 4](#_Toc121320231)

[1.1 Фнализ и описание предметной области 4](#_Toc121320232)

[1.2 Разработка концептуальной модели 5](#_Toc121320233)

[2 Определение требований к операционной системе 9](#_Toc121320234)

[3 Выбор субд и других программных средств 12](#_Toc121320235)

[4 Логическое проектирование реляционной базы данных 13](#_Toc121320236)

[5 Реализация проекта базы данных 29](#_Toc121320237)

[Заключение 37](#_Toc121320238)

[Список использованных источников 38](#_Toc121320239)

[Приложение А ­– Логическая моедель 39](#_Toc121320240)

[Приложение Б – Реалиционная модель 40](#_Toc121320241)

[Приложение В – Код создания таблиц 41](#_Toc121320242)

[Приложение Г – Код приложения 48](#_Toc121320243)

ВВЕДЕНИЕ

В настоящее время эффективное управление магазином бытовых товаров невозможно без грамотной организации и обработки данных. Для этих целей прекрасно подходит использование информационной системы, основанной на базе данных.

База данных для складского учета магазина бытовых товаров – это система, предназначенная для хранения, обработки и управления информацией о товарах, их количестве, поступлении, реализации и других сопутствующих параметрах. Введение такой системы в деятельность магазина позволяет оптимизировать процессы, улучшить точность учета и повысить общую эффективность.

Сфера управления складским учетом в магазинах бытовых товаров также требует внимания к эффективной обработке информации. Необходимо правильно использовать возможности для автоматизации процессов учета товаров, отслеживания поступлений и списаний, а также создания отчетов для эффективного управления складской деятельностью.

Внедрение баз данных в деятельность магазина создает возможность улучшить процессы учета и повысить общую эффективность. Таким образом, актуальной становится тема: "Технология разработки и защиты баз данных для складского учета магазина бытовых товаров".

Создание приложения для управления складским учетом даст возможность в любой момент времени просматривать информацию о товарах, редактировать, добавлять новые товары, удалять устаревшие записи, а также осуществлять фильтрацию данных для удобства работы с ассортиментом товаров.

Цель: разработать базу данных для интернета провайдера.

Задачи:

* произвести анализ и описать предметную область;
* разработать концептуальную и логическую модели; составить список сущностей и атрибутов, построить реляционную модель на основе логической модели;
* спроектировать базу данных с использованием средств SQL;
* разработать пользовательский интерфейс.

1 Инфологическое проектирование

1.1 Анализ и описание предметной области

Концептуальное проектирование – это построение модели предметной   
области, ориентированной на восприятие человека и независимой от типа системы управления базами данных (СУБД) и типа самой базы данных.

Под предметной областью (ПО) понимается совокупность связанных между собой функций, задач управления в некоторой области деятельности предприятия, с помощью которых достигается выполнение поставленной цели.

Множество фрагментов составляют из себя предметную область. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, с разными взглядами на ПО.

Модель предметной области представляет собой знания о предметной   
области. Знания могут быть неформальными и формальными. Модель предметной области описывает скорее процессы, происходящие в предметной области и   
данные, используемые этими процессами. От того, насколько правильно   
смоделирована предметная область, зависит успех дальнейшей разработки   
приложений.

Система создается для информационного обеспечения и учета деятельности преподавателей.

1.2 Разработка концептуальной модели

Для разработки базы данных необходимо определить основные сущности и необходимые поля. Так же следует выделить сопоставимые поля для определения связей между таблицами.

База данных должна содержать данные:

* о контрактах, подписанных с пользователями;
* о пользователях;
* о денежных транзакция, совершаемые пользователями;
* о местах подключения пользователей;
* о тарифах, доступных пользователям для подключения.

Учитывая все вышеперечисленное, база данных должна содержать следующие сущности и их атрибуты.

Сущность «Товары»:

* «Код\_товара»;
* «Код\_категории»;
* «Название»;
* «Цена».

Сущность «Категории\_товаров»:

* «Код\_категории»;
* «Название\_категории»;
* «Описание».

Сущность «Поставки»:

* «Код\_поставщиуа»;
* «Код\_товара»;
* «Дата\_поставки».

Сущность «Склады»:

* «Код\_склада»;
* «Название склада»;
* «Адрес».

Сущность «Товары\_на\_складе»:

* «Код\_товара»;
* «Код\_склада»;
* «Количество\_на\_складе».

Сущность «Постащики»:

* «Код\_Поставщика»;
* «Название\_компании»;
* «Адрес»;
* «Контактные\_данные».

Таким образом, были определены основные объекты будущей информационной системы.

Установим связи между сущностями и распишем их с помощью ERD-инструкции:

* «CUSTOMERS» и «CONTRACT», связь «один-ко-многим». Бизнес-правила:

1. каждый пользователь (CUSTOMERS) может иметь один или более контрактов (CONTRACT);
2. каждый контракт (CONTRACT) может иметь одного и только одного пользователя (CUSTOMERS).

* «MONETARY\_TRSNSACTION» и «CONTRACT», связь «один-ко-многим». Бизнес-правила:

1. каждая транзакиця (MONETARY\_TRSNSACTION) может содеджаться в одном или более контрактах (CONTRACT);
2. каждый контракт (CONTRACT) может содержать одну и только одну транзакцию (MONETARY\_TRSNSACTION)

* «PLACE» и «CONTRACT», связь «один-ко-многим». Бизнес-правила:

1. каждое место подключения (PLACE) может содержаться в одно или более контрактах (CONTRACT);
2. каждый контракт (CONTRACT) может содержать одно и только одно место подключения (PLACE).

* «TARIFF» и «CONTRACT», связь «один-ко-многим».

Бизнес-правила:

1. каждый тариф (TARIFF) может содержаться в одно или более контрактах(CONTRACT);
2. каждый контракт (CONTRACT) может содержать один и только один тариф (TARIFF).

При осуществлении нормализации текущих отношений было осуществлено приведение к нормальным формам:

* первая нормальная форма (1NF) — все атрибуты должны иметь только одно значение;
* вторая нормальная форма (2NF) — отношения находятся в первой нормальной форме и все не ключевые атрибуты зависят от первичного ключа;
* третья нормальная форма (3NF) — отношения находятся во второй нормальной форме и все неключевые атрибуты не транзитивно зависят от первичного ключа.

Нормализация – это процесс преобразования отношений базы данных к виду, отвечающему нормальным формам.

Нормализация предназначена для приведения структуры баз данных к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных.

Под избыточностью понимают повторение данных в разных строках одной таблицы или в разных таблицах БД. Итогом нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

В заключение главы выделим все выполненные в процессе создания концептуальной модели действия:

* были определены объекты базы данных и их атрибуты;
* были определены связи между объектами;
* были определены действия необходимые для приведения отношений в базе данных к нормальной форме.

2 Определение требований к операционной системе

По данной формуле будет вычисляться объем памяти, требуемый для хранения данных, на основании данных из таблицы 1:

𝑛

Мд = 2 ∑ 𝑙𝑖 ∗ (𝑁𝑖 + 𝑁𝑎𝑖 )

𝑖=1

где 𝑙𝑖 - длина записи в i-ой таблице (в байтах);

𝑁𝑖 - примерное (максимально возможное) количество записей в i-ой таблице;

𝑁𝑎 – количество записей в архиве i-ой таблицы;

*n* – количество таблиц в БД.

Таблица 2.1 – Атрибуты, отображающие длину записи в каждой таблице

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Отношения | Атрибут | Тип данных | Размер,  байт | Среднее  кол-во | Объем,  байт |
| CONTRACT | CUSTOMERS\_Number\_ID | Int | 2 | 50 | 6000 |
| Personal\_account\_number | int | 2 |
| TARIFF\_Tariff\_ID | int | 2 |
| Date\_start | date | 3 |
| Date\_end | date | 3 |
| Account\_balance | money | 3 |
| Contract\_status | varchar (1) | 1 |
| MONETARY\_TRSNSACTION\_ | int | 2 |
| PLACE\_Number\_ID | int | 2 |
| Login | varchar (50) | 50 |
| Password | varchar (50) | 50 |
| CUSTOMERS | ID | Int | 2 | 50 | 4050 |
| Personal\_code | varchar (5) | 5 |
| Full\_name | varchar (50) | 50 |

Продолжение таблицы 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Отношения | Атрибут | Тип данных | Размер,  байт | Среднее  кол-во | Объем,  байт |
|  | Gender | varchar (1) | 1 |  |  |
| Phone\_number | varchar (11) | 20 |
| MONETARY\_TRSNSACTION | ID | Int | 2 | 600 | 6000 |
| Client\_ID | Int | 2 |
| Transaction\_amount | Money | 3 |
| Date | Date | 3 |
| PLACE | ID | Int | 2 | 50 | 2900 |
| Street | Varchar (50) | 50 |
| House | Int | 2 |
| Corpus | Int | 2 |
| Flat | Int | 2 |
| TARIFF | ID | Int | 2 | 5 | 525 |
| Name\_tariff | Varchar (50) | 50 |
| Cost | Money | 3 |
| Description\_tariff | Varchar (50) | 50 |

Таким образом, объем памяти хранения данных за один год, составит:

МС = 2 × (6000 + 2900 + 525 + 6000 + 4050) = 19474 байт

Полученный объем памяти не будет являться постоянным, потому что он увеличивается или уменьшается по ходу работы с базой данных.

После проведенных вычислений становится ясно, что объем данных в информационной системе небольшой. Следовательно, для реализации проекта подойдет любой современный компьютер.

Объем оперативной памяти, который необходим, будет определяться требованиями выбранной системой управления баз данных, следовательно, СУБД MS SQL подходит для введения данной базы данных.

Для поддержания целостности данных выполняется резервное копирование баз данных. План резервного копирования приведен в таблице 2.2.

Таблица 2.2 – План резервного копирования

|  |  |
| --- | --- |
| День недели | Вид резервного копирования |
| Понедельник | Разностное резервное копирование (контрактов и транзакций) |
| Вторник | Разностное резервное копирование (контрактов и транзакций) |
| Среда | Разностное резервное копирование (контрактов и транзакций) |
| Четверг | Разностное резервное копирование (контрактов и транзакций) |
| Пятница | Разностное резервное копирование (контрактов и транзакций) |
| Суббота | Разностное резервное копирование (контрактов и транзакций) |
| Воскресенье | Полное резервное копирование всей БД |

В Microsoft SQL Server принята практика разных типов резервного копирования. Пользователям доступно:

* полное – делается резервная копия всей БД;
* дифференциальное или разностное – осуществляется копирование данных с того момента, когда осуществлялось ее последнее полное резервирование;
* логов или инкрементальное.

3 Выбор СУБД и других программных средств

При создании базы данных и клиентского приложения были решены важные и сложные задачи, для успешного выполнения которых нужно было правильно сделать выбор средства реализации проекта. Для создания концептуальной и логической моделей базы данных Oracle SQL Developer Data Modeler предоставил много возможностей. Приложение является бесплатным графическим программным обеспечением, с помощью которого можно сконструировать различные модели данных, такие как логические, реляционные, многомерные, сетевые и т. д.

СУБД MS SQL Server стала отличным выбором для создания, использования и управления будущей базы данных. Обладая не только всеми необходимыми функциями для создания клиентского приложения, но и имея высокую производительность, она стала самым удачным выбором, так как другие приложения компании Microsoft могут работать с ней без каких-либо проблем.

Следовательно, наилучшим вариантом после выбора СУБД стала программная среда Microsoft Visual Studio с использованием объектно-ориентированного языка программирования C# для разработки пользовательского интерфейса и программного кода, и комфортной работы пользователей с базой данных.

Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows.

Язык программирования C# был выбран из-за его перспективности среди других языков и обстоятельств, облегчающих дальнейшее сопровождение программного обеспечения.

4 Логическое проектирование реляционной базы данных

Логическое проектирование является созданием схемы баз данных на основе конкретной модели данных, то есть в данном случае реляционной модели данных, представленных на рисунках 4.1 и 4.2.

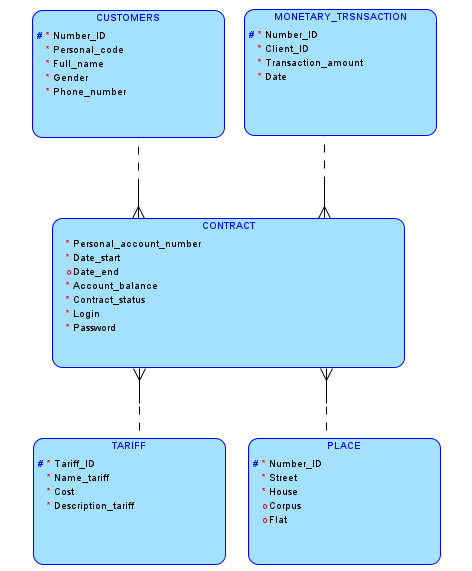


Рисунок 4.1 – Логическая модель базы данных

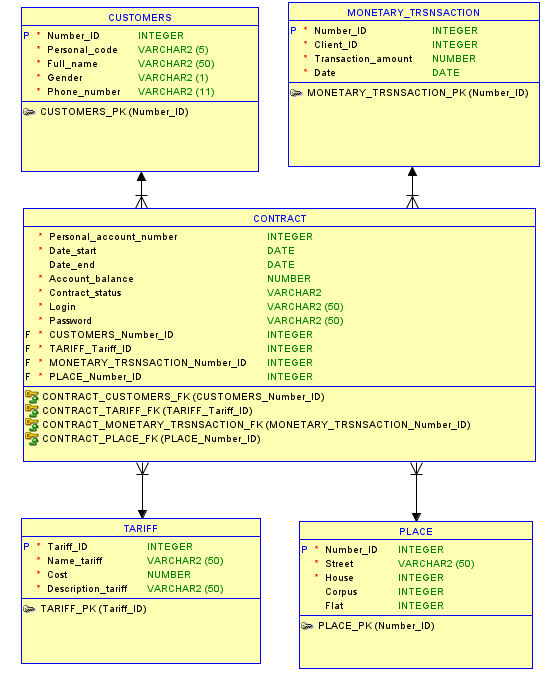


Рисунок 4.2 – Реляционная модель базы данных

Сущностью в базе данных может являться практически любой объект, который возможно логически выделить из предметной области, а ее атрибутом именованный столбец таблицы. Особыми атрибутами считаются первичные и внешние ключи.

Первичный ключ – это столбец, значения которого однозначно определяет строку.

Внешний ключ – это столбец в данной таблице, значения которого совпадает со значениями столбца, являющийся первичным ключем в другой таблице.

Во время разработки структуры базы данных были созданы таблицы, представления, хранимые процедуры и триггеры.

В базе данных таблица является совокупностью, связанной между собой информации, хранящейся в определенном порядке. Любая таблица состоит из кортежей (строк) и атрибутов (столбцов).

Представленная база данных состоит из шести таблиц:

* Таблица «Товары»;
* Таблица «Категории\_товаров»;
* Таблица «Склады»;
* Таблица «Поставщики»;
* Таблица «Товары\_на\_складе»;
* Таблица «Поставки».

Таблица «CONTRACT» содержит информацию о контрактах: их пользователей и данные, связанные с их тарифом, местом подключения и прочими. Таблица «CUSTOMERS» представляет собой совокупность информации о пользователях, а таблица «PLACE» – место подключения. Таблица «MONETARY\_TRSNSACTION» хранит данные о проведенных транзакция. В таблице «TARIFF» собраны сведения о доступных тарифах для подключения.

Все таблицы созданы в одной базе данных. Описание структуры таблиц и их полей приведены в таблицах 4.1 – 4.5.

Таблица 4.1 – Структура таблицы «Товары»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Personal\_account\_number | Int | 2 | Идентифицирующий  номер аккаунта | Первичный  ключ |
| Date\_start | Date | 3 | Дата начала | NOT NULL |
| Date\_end | Date | 3 | Дата конца |  |
| Account\_balance | Money | 5 | Баланс аккаунта | NOT NULL |
| Contract\_status | Varchar(1) | 1 | Статус контракта | NOT NULL |
| Login | Varchar(50) | 50 | Логин | NOT NULL |
| Password | Varchar(50) | 50 | Пароль | NOT NULL |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| CUSTOMERS\_Number\_ID | Int | 2 | Идентифицирующий  номер пользователя | Внешний ключ |
| TARIFF\_Tariff\_ID | Int | 2 | Идентифицирующий  номер пользователя | Внешний ключ |
| MONETARY\_TRSNSACTION\_Number\_ID | Int | 2 | Идентифицирующий  номер транзакции | Внешний ключ |
| PLACE\_Number\_ID | Int | 2 | Идентифицирующий  номер места подключения | Внешний ключ |

Таблица 4.2 – Структура таблицы «CUSTOMERS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Number\_ID | Int | 2 | Идентифицирующий  номер пользователя | Первичный  ключ |
| Personal\_code | Varchar(5) | 5 | Персональный код | NOT NULL |
| Full\_name | Varchar(50) | 50 | ФИО | NOT NULL |
| Gender | Varchar(1) | 1 | Пол | NOT NULL |
| Phone\_number | Varchar(11) | 11 | Номер телефона | NOT NULL |

Таблица 4.3 – Структура таблицы «MONETARY\_TRSNSACTION»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Number\_ID | Int | 2 | Идентифицирующий  номер транзакции | Первичный  ключ |
| Client\_ID | Int | 2 | Идентифицирующий  номер клиента | NOT NULL |
| Transaction\_amount | Money | 5 | Сумма транзакции | NOT NULL |
| Date | Date | 3 | Дата | NOT NULL |

Таблица 4.4 – Структура таблицы «PLACE»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Number\_ID | Int | 2 | Идентифицирующий  номер места подключения | Первичный  ключ |
| Street | Varchar(50) | 50 | Название улицы | NOT NULL |
| House | Int | 2 | Дом | NOT NULL |
| Сorpus | Int | 2 | Корпус |  |
| Flat | Int | 2 | Квартира |  |

Таблица 4.5 – Структура таблицы «TARIFF»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Tariff\_ID | Int | 2 | Идентифицирующий  номер тарифа | Первичный  ключ |
| Name\_tariff | Varchar(50) | 50 | Название тарифа | NOT NULL |
| Cost | Money | 2 | Стоимость | NOT NULL |
| Description\_tariff | Varchar(50) | 50 | Описание тарифа | NOT NULL |

После создания таблиц и добавления данных в MS SQL Server база данных выглядит следующим образом рисунок 4.3, рисунок 4.4, рисунок 4.5.

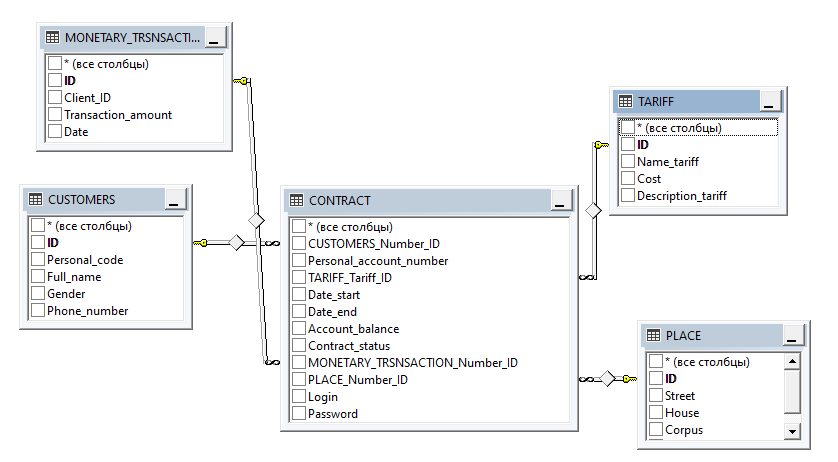


Рисунок 4.3 – Диаграмма базы данных

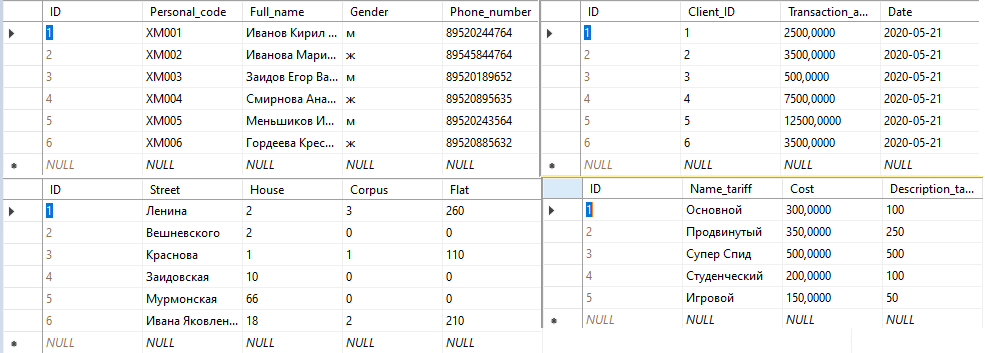


Рисунок 4.4 – Таблицы базы данных

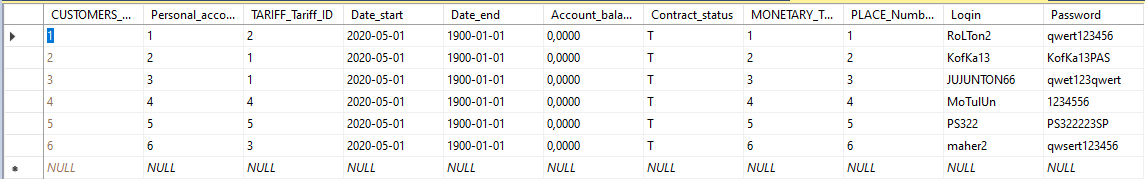


Рисунок 4.5 – Продолжение таблиц базы данных

Представление по своему внешнему виду напоминает таблицу, но на самом деле является не самостоятельным объектом, а результатом запроса на выборку к существующей в базе данных таблице и полностью зависит от нее, так как само по себе представление не может содержать никаких данных.

Созданные представления в данном проекте выглядят следующим образом:

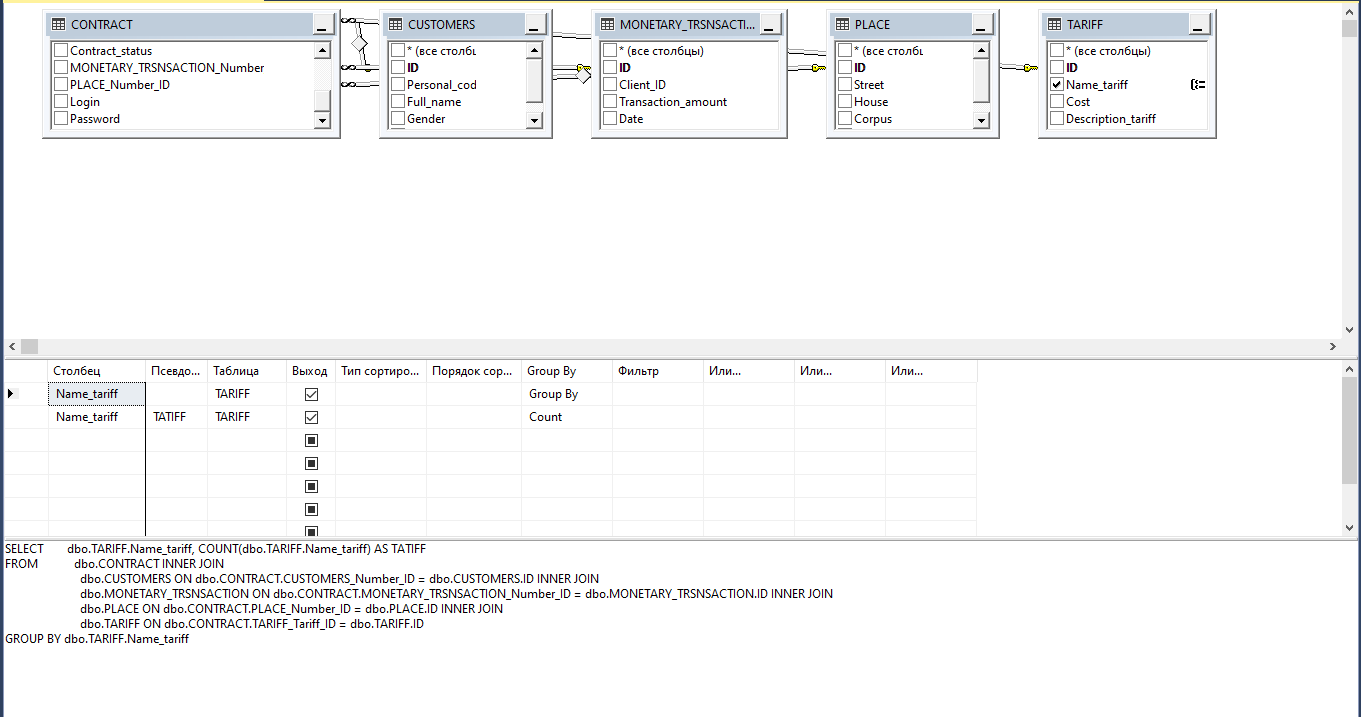


Рисунок 4.5 – Представление «Tariff Count»

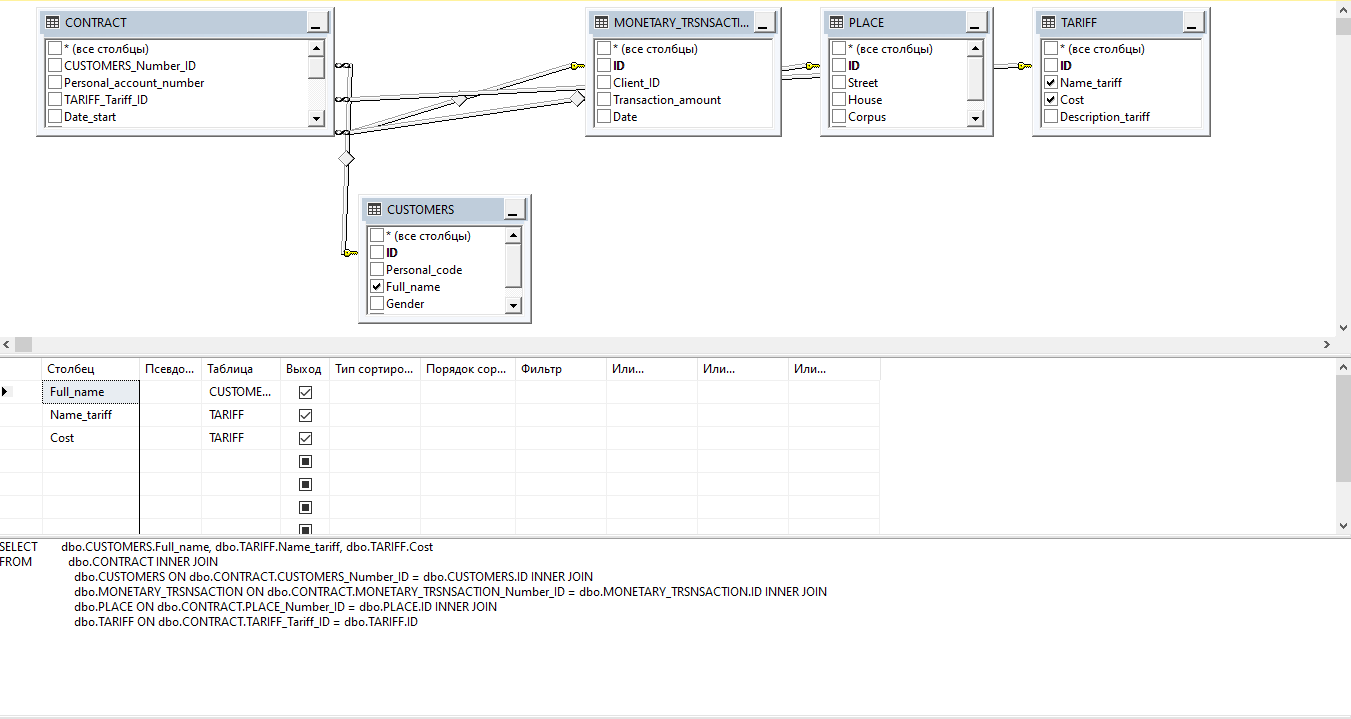


Рисунок 4.6 – Представление «User and Tariff»

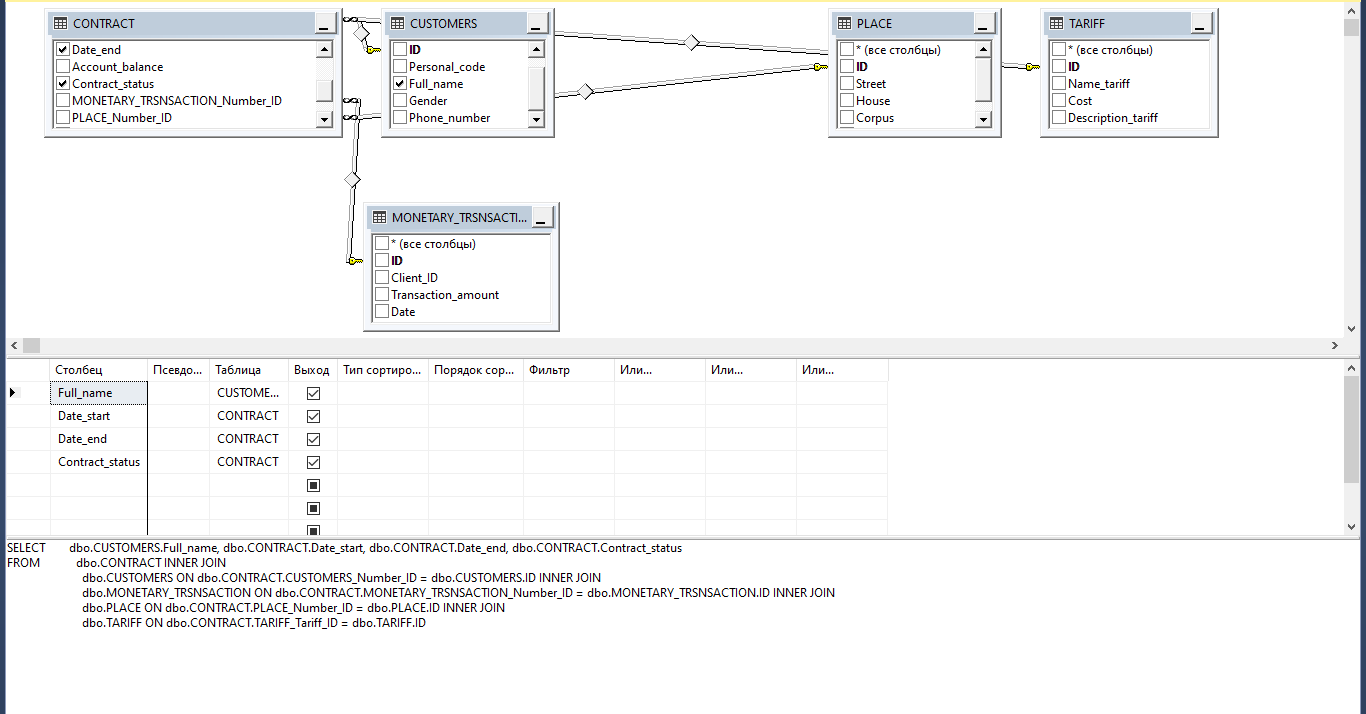


Рисунок 4.7 – Представление «User Contract»

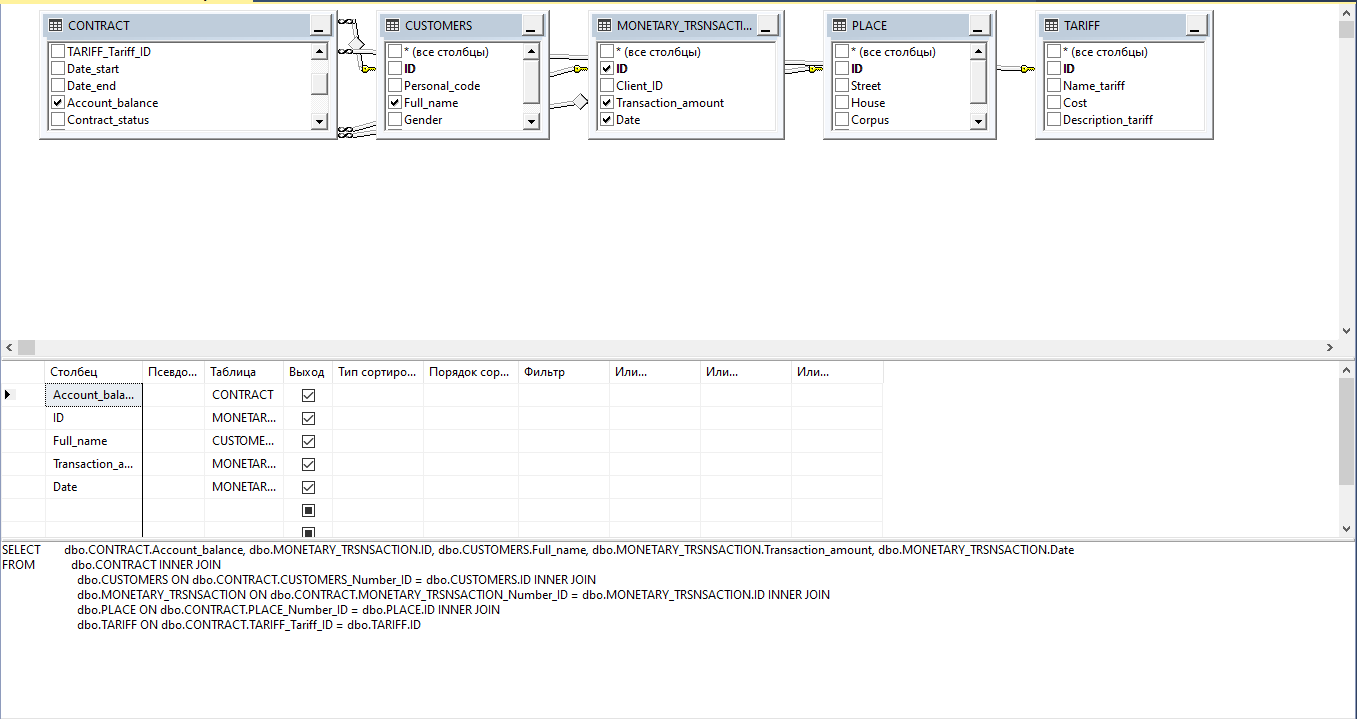


Рисунок 4.8 – Представление «User Money»

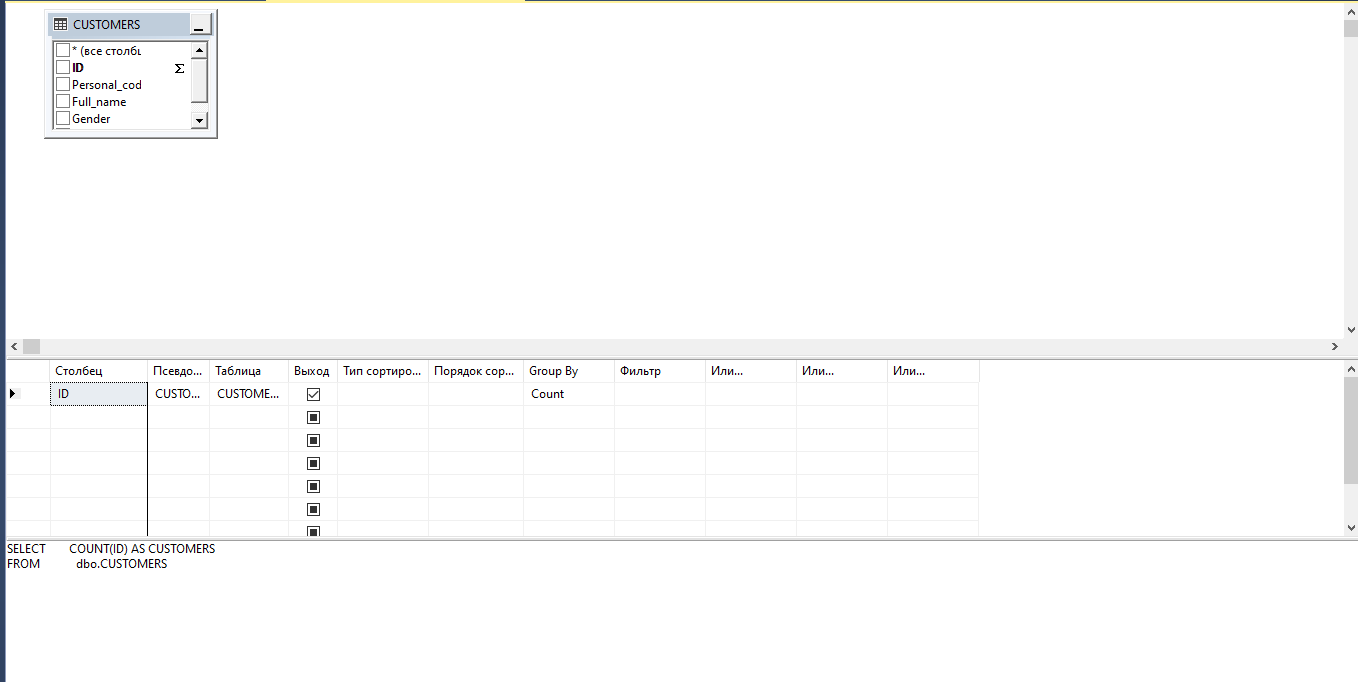


Рисунок 4.9 – Представление «User in Date base»

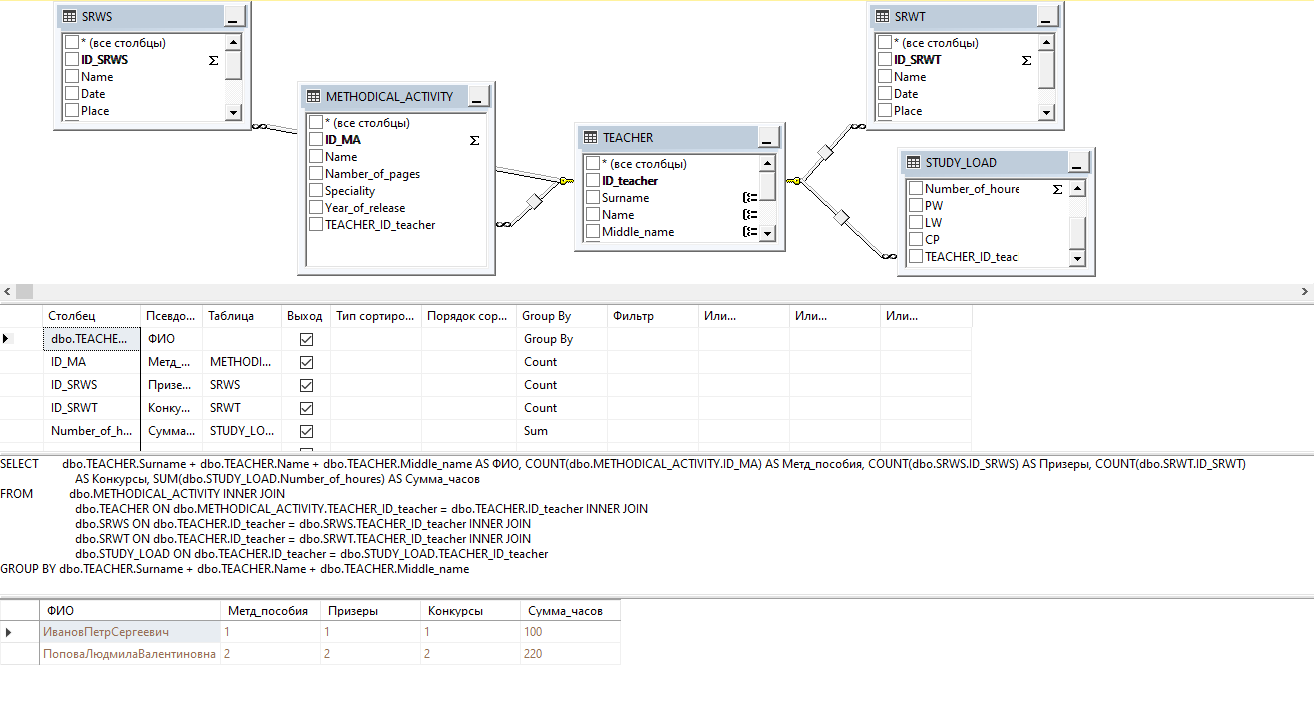


Рисунок 4.10 – Представление «Преподаватели»

Хранимая процедура – это набор команд, состоящий из одного или нескольких операторов SQL, сохраняющий в БД в откомпилированном виде. Такие процедуры могут иметь входные и выходные параметры, а также локальные переменные.

При разработке курсового проекта были созданы следующие хранимые процедуры:

* Update\_Contract рисунок 4.12. Данная хранимая процедура позволяет обновлять данные в таблице CONTRACT.

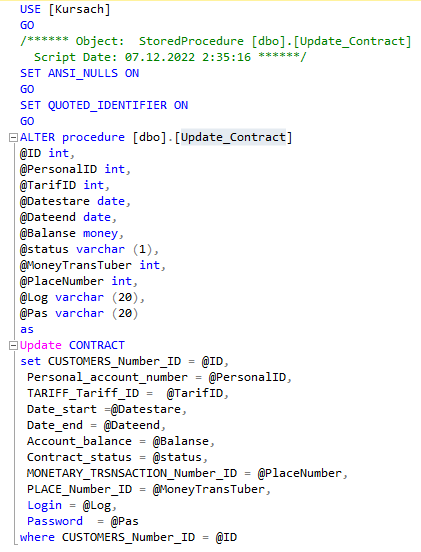


Рисунок 4.12 – Скрипт на создание хранимой процедуры Update\_Contract

* Insert\_Contract рисунок 4.13. Данная хранимая процедура позволяет добавлять данные в таблицу CONTRACT.

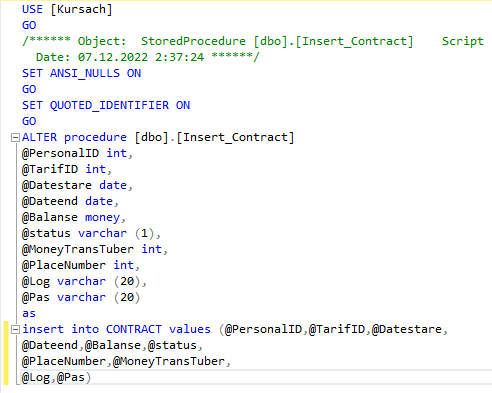


Рисунок 4.13 – Скрипт на создание хранимой процедуры Insert\_Contract

* Delete\_Contract рисунок 4.14. Данная хранимая процедура позволяет удалить выбранную запись в таблице CONTRACT.

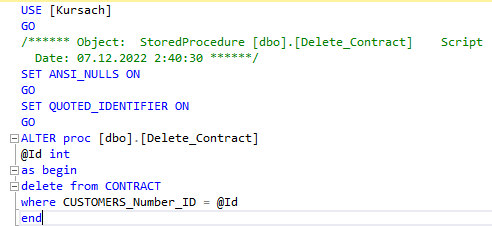


Рисунок 4.14 – Скрипт на создание хранимой процедуры Delete\_Contract

Insert\_Customers рисунок 4.15. Данная хранимая процедура позволяет добавить новую запись в таблицу CUSTMERS.

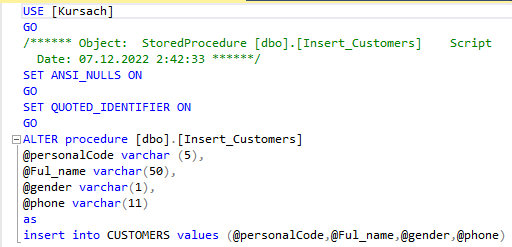


Рисунок 4.15 – Скрипт на создание хранимой процедуры Insert\_Customers

Update\_Customers рисунок 4.16. Данная хранимая процедура позволяет обновить запись в таблицу CUSTMERS.



Рисунок 4.16 – Скрипт на создание хранимой процедуры Update\_Customers

Delete\_Customers рисунок 4.17. Данная хранимая процедура позволяет удалить из таблицы CUSTOMERS.

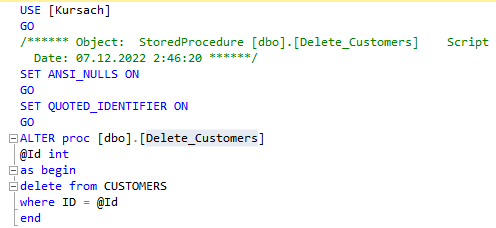


Рисунок 4.17 – Скрипт на создание хранимой процедуры Delete\_Customers

Триггер – это вид хранимой процедуры, выполняющийся автоматически при возникновении событий языка обработки данных в БД. Триггеры используются для проверки целостности данных, а также для отката транзакций. Существует несколько видов: вставка, удаление или редактирование информации   
в определенной таблице или во всей базе данных.

Также в курсовом проекте были созданы триггеры:

* Control\_balance, реализующий добавление суммы транзакции в баланс, код создания триггера представлен на рисунке 4.18.

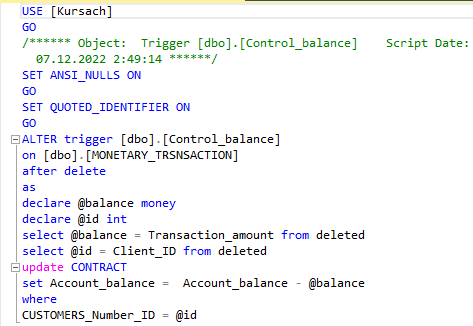


Рисунок 4.18 – Скрипт на создание триггера Control\_balance

* Date\_Control, не позволяющий производить транзакцию раньше сегодяшней даты, код данного триггера показан на рисунке 4.19.

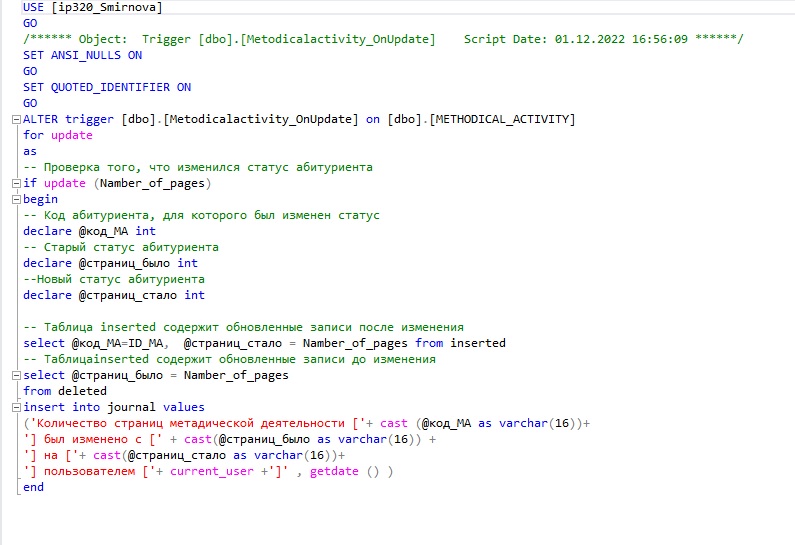


Рисунок 4.19 – Скрипт на создание триггера Date\_Control

* Update\_balance, реализация контроля за балансом, путем обновления баланса после редактрирования или удаления транзакции рисунок 4.20

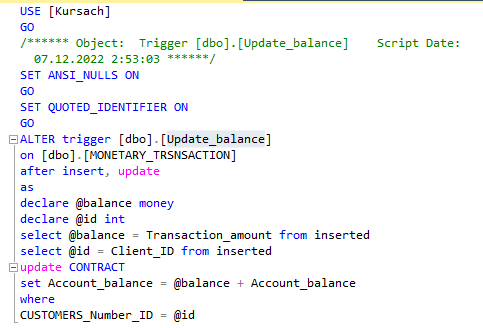


Рисунок 4.20 – Скрипт на создание триггера Update\_balance

Фрагменты создания таблиц, представлений, хранимых процедур и триггеров приведены в приложении А.

5 Реализация проекта базы данных

Для полного комфорта и удобства пользователя при работе с базой данных было создано пользовательское приложение, представляющее собой форму Windows Forms, созданную в среде разработки Microsoft Visual Studio 2019.

Подключение к базе данных осуществляется с помощью технологии ADO.NET Entity Framework, то есть объектно-ориентированной технологией доступа к данным, являясь object-relational mapping решением для .NET Framework от Microsoft.

При запуске приложения отрывается окно авторизации рисунок 5.1, в котором предлагается вести свой логин и пароль. Если вводится неправильный пароль, появляется окно с оповещением рисунок 5.2.

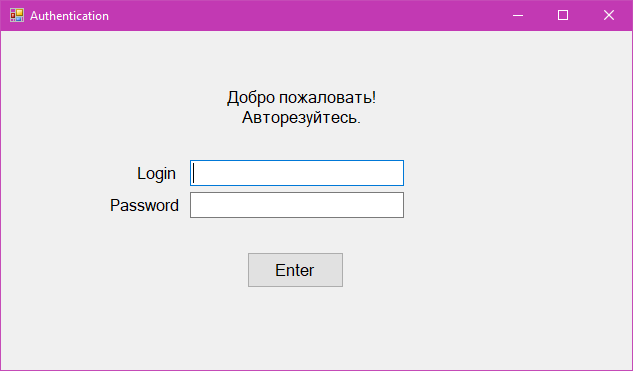


Рисунок 5.1 – Окно авторизации

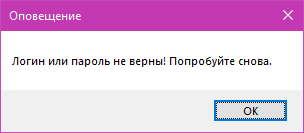


Рисунок 5.2 – Сообщение об ошибке

Если пароль указан верно, то происходит переход на следующую форму рисунок 5.3.



Рисунок 5.3 – Окно работы с таблицами

Переход по существующим таблицам происходит через форму «Menu» с помощью кнопок рисунок 5.4, рисунок 5.5, рисунок 5.6, рисунок 5.7, рисунок 5.8, рисунок 5.9, рисунок 5.10.

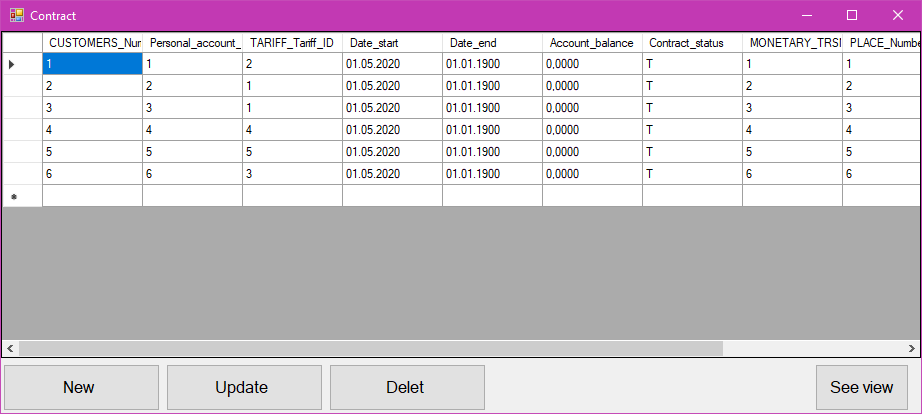


Рисунок 5.4 – Таблица «CONTRACT»

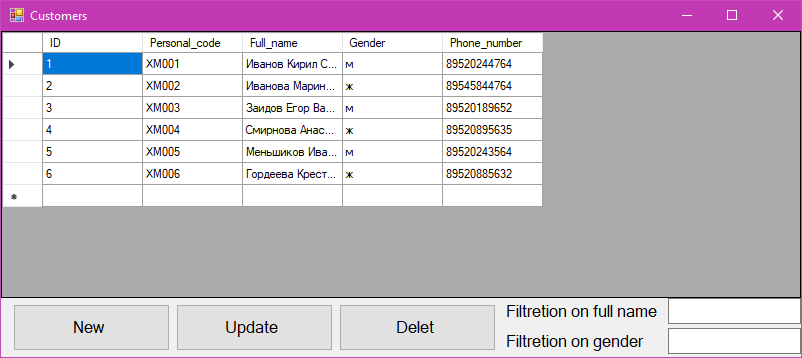


Рисунок 5.5 – Таблица «CUSTOMERS»

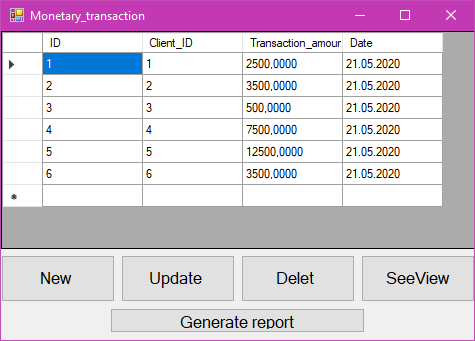


Рисунок 5.6 – Таблица «MONETARY\_TRANSACTION»

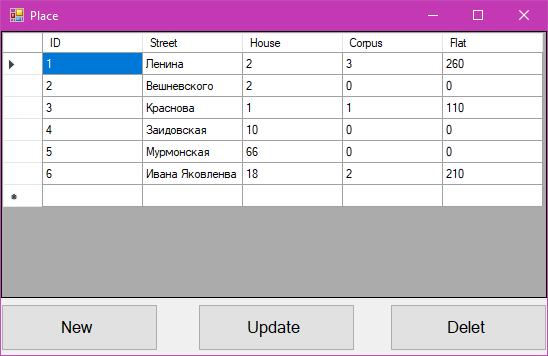


Рисунок 5.7 – Таблица «PLACE

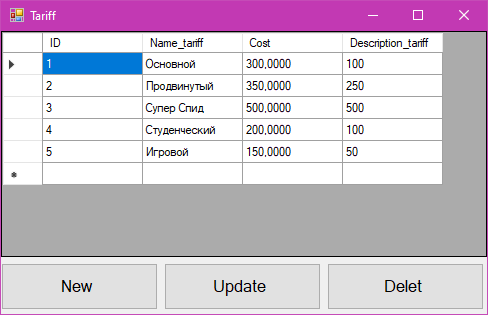


Рисунок 5.8 – Таблица «TARIFF»

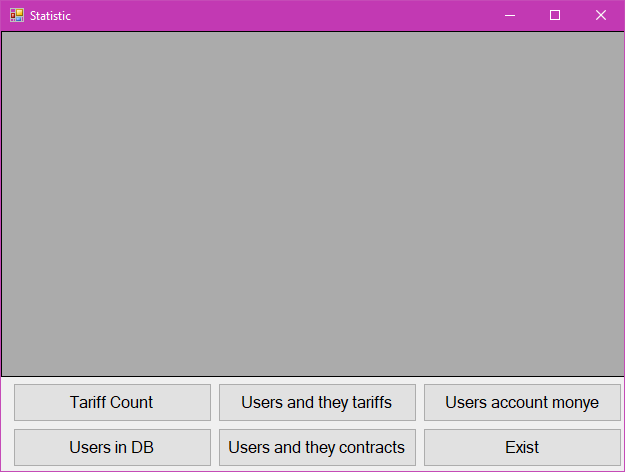


Рисунок 5.9 – Таблица «STATISTIC»

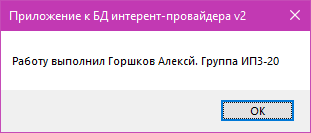


Рисунок 5.10 – Сведения об разработчике

Кнопки «New», «Update» и «Delete» вызывают исполнение соответствующих команд. При нажатии на первую или третью кнопку открывается специальный диалог редактирования таблицы рисунок 5.11, 5.12.

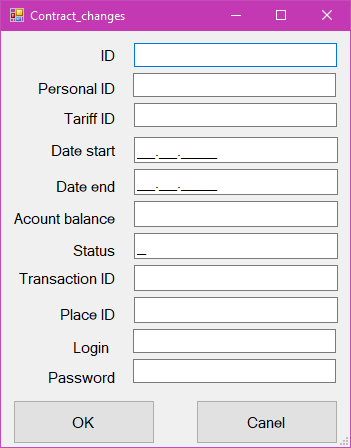


Рисунок 5.11 – Окно добавления таблицы «CONTRACT»

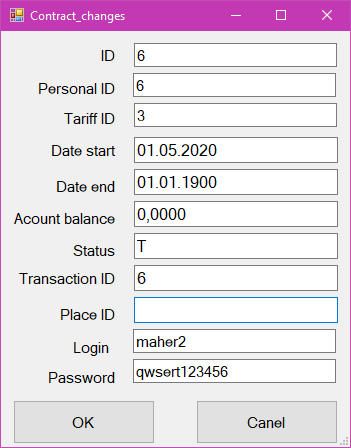


Рисунок 5.12 – Окно редактирования таблицы «CONTRACT»

Фильтрация осуществленна по определенным параметрам рисунок 5.13.

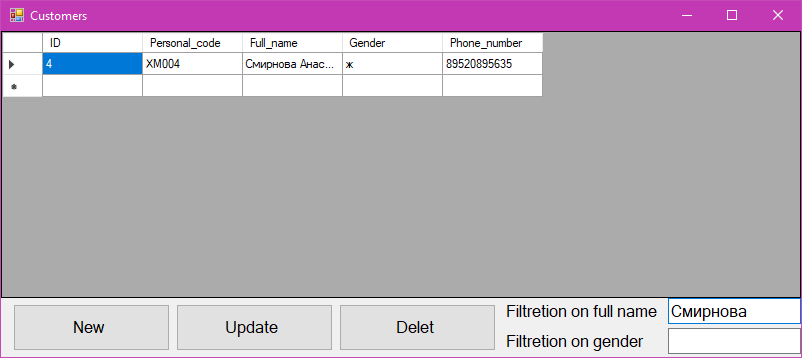


Рисунок 5.13 – Фильтрация на форме «CUSTOMERS»

На форме «Statistic» реализован вывод представлений для аналитики рисунок 5.14.

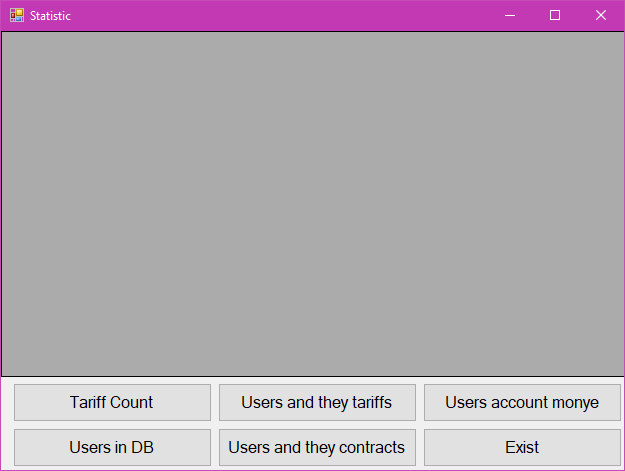


Рисунок 5.14 – Фильтрация на форме «Metodical\_activity»

Кнопка «Generate reprs» на форме «MONETARY\_TRANSACTION» создает отчет в Microsoft Excel рисунок 5.15

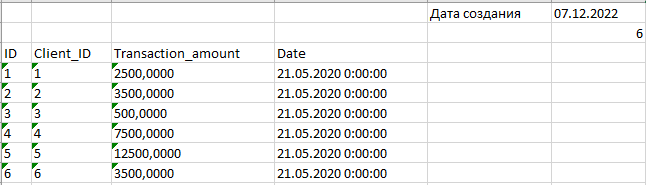


Рисунок 5.15 – Отчет в Microsoft Excel

По итогу результатом пройденного пути стало создание приложения, полностью удовлетворяющим все требования. Данное приложение позволяет свободно работать с созданной ранее БД.

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта были решены все поставленные задачи:

* были произведены анализ и описание предметной области;
* разработаны концептуальная и логическая модель базы данных;
* спроектирована база данных с использованием средств SQL;
* разработан пользовательский интерфейс.

А также получены новые навыки по работе с такими программами как: Oracle SQL Developer Data Modeler, Microsoft Visual Studio 2019 и СУБД Microsoft SQL Server.

Итоговое приложение полностью удовлетворяет запросы, так как позволяет свободно вносить правки в базу данных, создавать информацию о новых контрактах, пользователях, местах подключения, тарифах и денежных транзакциях. Программный продукт способен создавать итоговые отчеты и обрабатывать большое количество информации.

После решения всех поставленных задач была достигнута цель курсового проекта – разработана база данных интернет-провайдера и ее защита.

Можно и дальше улучшать приложение, например, создание следующий версии исходя из приобретенного опыта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Федорова, Г. Н. Разработка, администрирование и защита баз данных: учебник для студ. учреждений сред. проф. Образования / Федорова Г. Н. – Москва: Издательский центр «Академия», 2019. – 288 с.
2. Агальцов, В. П. Базы данных: в 2 книгах. Книга 2. Распределенные и удаленные базы данных: учебник / В.П. Агальцов. — Москва: ФОРУМ: ИНФРА- М, 2021. — 271 с. — URL: https://znanium.com/catalog/product/1514118 (дата обращения: 04.12.2022).
3. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. — Москва: ФОРУМ: ИНФРА-М, 2021. — 368 с. — URL: https://znanium.com/catalog/product/1215513 (дата обращения: 04.12.2022).

## ПРИЛОЖЕНИЕ А

Логическая модель

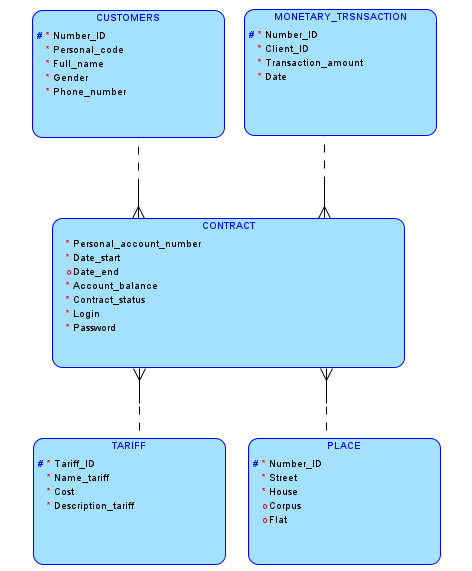


Рисунок А.1 – Логическая модель

# ПРИЛОЖЕНИЕ Б

Реалиционная модель

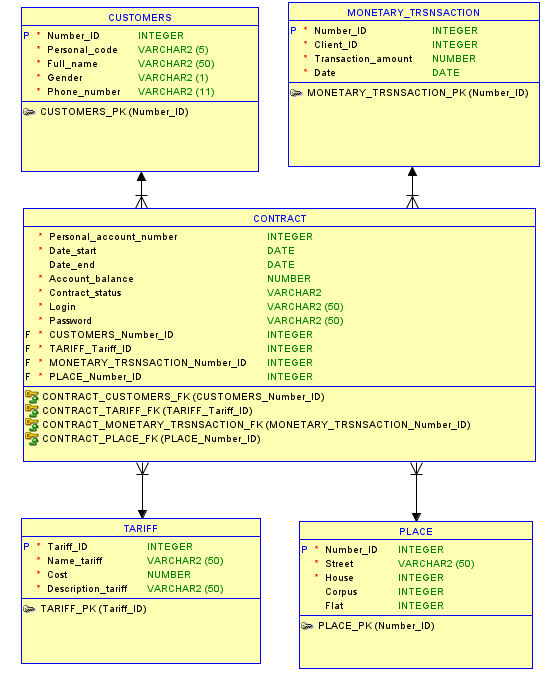


Рисунок Б.1 – Логическая модель

ПРИЛОЖЕНИЕ B

Код создания таблиц

CREATE TABLE CONTRACT

(

CUSTOMERS\_Number\_ID INTEGER NOT NULL ,

Personal\_account\_number INTEGER NOT NULL ,

TARIFF\_Tariff\_ID INTEGER NOT NULL ,

Date\_start DATE NOT NULL ,

Date\_end DATE ,

Account\_balance MONEY NOT NULL ,

Contract\_status VARCHAR NOT NULL ,

MONETARY\_TRSNSACTION\_Number\_ID INTEGER NOT NULL ,

PLACE\_Number\_ID INTEGER NOT NULL ,

Login VARCHAR (50) NOT NULL ,

Password VARCHAR (50) NOT NULL

)

GO

CREATE TABLE CUSTOMERS

(

ID INTEGER NOT NULL ,

Personal\_code VARCHAR (5) NOT NULL ,

Full\_name VARCHAR (50) NOT NULL ,

Gender VARCHAR (1) NOT NULL ,

Phone\_number VARCHAR (11) NOT NULL

)

GO

ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS\_PK PRIMARY KEY CLUSTERED (ID)

WITH (

ALLOW\_PAGE\_LOCKS = ON ,

ALLOW\_ROW\_LOCKS = ON )

GO

CREATE TABLE MONETARY\_TRSNSACTION

(

ID INTEGER NOT NULL ,

Client\_ID INTEGER NOT NULL ,

Transaction\_amount MONEY NOT NULL ,

Date DATE NOT NULL

)

GO

ALTER TABLE MONETARY\_TRSNSACTION ADD CONSTRAINT MONETARY\_TRSNSACTION\_PK PRIMARY KEY CLUSTERED (ID)

WITH (

ALLOW\_PAGE\_LOCKS = ON ,

ALLOW\_ROW\_LOCKS = ON )

GO

CREATE TABLE PLACE

(

ID INTEGER NOT NULL ,

Street VARCHAR (50) NOT NULL ,

House INTEGER NOT NULL ,

Сorpus INTEGER ,

Flat INTEGER

)

GO

ALTER TABLE PLACE ADD CONSTRAINT PLACE\_PK PRIMARY KEY CLUSTERED (ID)

WITH (

ALLOW\_PAGE\_LOCKS = ON ,

ALLOW\_ROW\_LOCKS = ON )

GO

CREATE TABLE TARIFF

(

ID INTEGER NOT NULL ,

Name\_tariff VARCHAR (50) NOT NULL ,

Cost MONEY NOT NULL ,

Description\_tariff VARCHAR (50) NOT NULL

)

GO

ALTER TABLE TARIFF ADD CONSTRAINT TARIFF\_PK PRIMARY KEY CLUSTERED (ID)

WITH (

ALLOW\_PAGE\_LOCKS = ON ,

ALLOW\_ROW\_LOCKS = ON )

GO

ALTER TABLE CONTRACT

ADD CONSTRAINT CONTRACT\_CUSTOMERS\_FK FOREIGN KEY

(

CUSTOMERS\_Number\_ID

)

REFERENCES CUSTOMERS

(

ID

)

ON DELETE NO ACTION

ON UPDATE NO ACTION

GO

ALTER TABLE CONTRACT

ADD CONSTRAINT CONTRACT\_MONETARY\_TRSNSACTION\_FK FOREIGN KEY

(

MONETARY\_TRSNSACTION\_Number\_ID

)

REFERENCES MONETARY\_TRSNSACTION

(

ID

)

ON DELETE NO ACTION

ON UPDATE NO ACTION

GO

ALTER TABLE CONTRACT

ADD CONSTRAINT CONTRACT\_PLACE\_FK FOREIGN KEY

(

PLACE\_Number\_ID

)

REFERENCES PLACE

(

ID

)

ON DELETE NO ACTION

ON UPDATE NO ACTION

GO

ALTER TABLE CONTRACT

ADD CONSTRAINT CONTRACT\_TARIFF\_FK FOREIGN KEY

(

TARIFF\_Tariff\_ID

)

REFERENCES TARIFF

(

ID

)

insert into CUSTOMERS values

('XM001','Иванов Кирил Степаныч','м','89520244764'),

('XM002','Иванова Марина Степановна','ж','89545844764'),

('XM003','Заидов Егор Вагифович','м','89520189652'),

('XM004','Смирнова Анастасия Александрова','ж','89520895635'),

('XM005','Меньшиков Ивано Алексеевич','м','89520243564'),

('XM006','Гордеева Крестина Витальевна','ж','89520885632')

insert into MONETARY\_TRSNSACTION values

('001',2500,'21-05-2020'),

('002',3500,'21-05-2020'),

('003',500,'21-05-2020'),

('004',7500,'21-05-2020'),

('005',12500,'21-05-2020'),

('006',350,'21-05-2020')

insert into PlACE values

('Ленина','2','3','260'),

('Вешневского','2','',''),

('Краснова','1','1','110'),

('Заидовская','10','',''),

('Мурмонская','66','',''),

('Ивана Яковленва','18','2','210')

insert into TARIFF values

('Базовый','300','100'),

('Продвинутый','350','250'),

('Супер Спид','500','500'),

('Студенческий','200','100'),

('Игровой','150','50')

insert into CONTRACT values

(1,1,2,'01-05-2020','','','T',1,1,'RoLTon2','qwert123456'),

(2,2,1,'01-05-2020','','','T',2,2,'KofKa13','KofKa13PAS'),

(3,3,1,'01-05-2020','','','T',3,3,'JUJUNTON66','qwet123qwert'),

(4,4,4,'01-05-2020','','','T',4,4,'MoTulUn','1234556'),

(5,5,5,'01-05-2020','','','T',5,5,'PS322','PS322223SP'),

(6,6,3,'01-05-2020','','','T',6,6,'maher2','qwsert123456')

Код создания представлений:

CREATE VIEW Tariff\_count

SELECT SELECT dbo.TARIFF.Name\_tariff, COUNT(dbo.TARIFF.Name\_tariff) AS TATIFF

FROM dbo.CONTRACT INNER JOIN

dbo.CUSTOMERS ON dbo.CONTRACT.CUSTOMERS\_Number\_ID = dbo.CUSTOMERS.ID INNER JOIN

dbo.MONETARY\_TRSNSACTION ON dbo.CONTRACT.MONETARY\_TRSNSACTION\_Number\_ID = dbo.MONETARY\_TRSNSACTION.ID INNER JOIN

dbo.PLACE ON dbo.CONTRACT.PLACE\_Number\_ID = dbo.PLACE.ID INNER JOIN

dbo.TARIFF ON dbo.CONTRACT.TARIFF\_Tariff\_ID = dbo.TARIFF.ID

GROUP BY dbo.TARIFF.Name\_tariff

CREATE VIEW User\_and\_tariff

SELECT dbo.TARIFF.Name\_tariff, COUNT(dbo.TARIFF.Name\_tariff) AS TATIFF

FROM dbo.CONTRACT INNER JOIN

dbo.CUSTOMERS ON dbo.CONTRACT.CUSTOMERS\_Number\_ID = dbo.CUSTOMERS.ID INNER JOIN

dbo.MONETARY\_TRSNSACTION ON dbo.CONTRACT.MONETARY\_TRSNSACTION\_Number\_ID = dbo.MONETARY\_TRSNSACTION.ID INNER JOIN

dbo.PLACE ON dbo.CONTRACT.PLACE\_Number\_ID = dbo.PLACE.ID INNER JOIN

dbo.TARIFF ON dbo.CONTRACT.TARIFF\_Tariff\_ID = dbo.TARIFF.ID

GROUP BY dbo.TARIFF.Name\_tariff

CREATE VIEW User\_contract

SELECT dbo.CUSTOMERS.Full\_name, dbo.CONTRACT.Date\_start, dbo.CONTRACT.Date\_end, dbo.CONTRACT.Contract\_status

FROM dbo.CONTRACT INNER JOIN

dbo.CUSTOMERS ON dbo.CONTRACT.CUSTOMERS\_Number\_ID = dbo.CUSTOMERS.ID INNER JOIN

dbo.MONETARY\_TRSNSACTION ON dbo.CONTRACT.MONETARY\_TRSNSACTION\_Number\_ID = dbo.MONETARY\_TRSNSACTION.ID INNER JOIN

dbo.PLACE ON dbo.CONTRACT.PLACE\_Number\_ID = dbo.PLACE.ID INNER JOIN

dbo.TARIFF ON dbo.CONTRACT.TARIFF\_Tariff\_ID = dbo.TARIFF.ID

CREATE VIEW User\_money

SELECT dbo.CONTRACT.Account\_balance, dbo.MONETARY\_TRSNSACTION.ID, dbo.CUSTOMERS.Full\_name, dbo.MONETARY\_TRSNSACTION.Transaction\_amount, dbo.MONETARY\_TRSNSACTION.Date

FROM dbo.CONTRACT INNER JOIN

dbo.CUSTOMERS ON dbo.CONTRACT.CUSTOMERS\_Number\_ID = dbo.CUSTOMERS.ID INNER JOIN

dbo.MONETARY\_TRSNSACTION ON dbo.CONTRACT.MONETARY\_TRSNSACTION\_Number\_ID = dbo.MONETARY\_TRSNSACTION.ID INNER JOIN

dbo.PLACE ON dbo.CONTRACT.PLACE\_Number\_ID = dbo.PLACE.ID INNER JOIN

dbo.TARIFF ON dbo.CONTRACT.TARIFF\_Tariff\_ID = dbo.TARIFF.ID

CREATE VIEW User\_in\_DB

SELECT COUNT(ID) AS CUSTOMERS

FROM dbo.CUSTOMERS

Код создания хранимых процедур:

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Delete\_Contract] Script Date: 07.12.2022 3:22:12 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER proc [dbo].[Delete\_Contract]

@Id int

as begin

delete from CONTRACT

where CUSTOMERS\_Number\_ID = @Id

end

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Update\_Contract] Script Date: 07.12.2022 3:22:28 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER procedure [dbo].[Update\_Contract]

@ID int,

@PersonalID int,

@TarifID int,

@Datestare date,

@Dateend date,

@Balanse money,

@status varchar (1),

@MoneyTransTuber int,

@PlaceNumber int,

@Log varchar (20),

@Pas varchar (20)

as

Update CONTRACT

set CUSTOMERS\_Number\_ID = @ID,

Personal\_account\_number = @PersonalID,

TARIFF\_Tariff\_ID = @TarifID,

Date\_start =@Datestare,

Date\_end = @Dateend,

Account\_balance = @Balanse,

Contract\_status = @status,

MONETARY\_TRSNSACTION\_Number\_ID = @PlaceNumber,

PLACE\_Number\_ID = @MoneyTransTuber,

Login = @Log,

Password = @Pas

where CUSTOMERS\_Number\_ID = @ID

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Insert\_Contract] Script Date: 07.12.2022 3:22:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER procedure [dbo].[Insert\_Contract]

@PersonalID int,

@TarifID int,

@Datestare date,

@Dateend date,

@Balanse money,

@status varchar (1),

@MoneyTransTuber int,

@PlaceNumber int,

@Log varchar (20),

@Pas varchar (20)

as

insert into CONTRACT values (@PersonalID,@TarifID,@Datestare,@Dateend,@Balanse,@status,@PlaceNumber,@MoneyTransTuber,@Log,@Pas)

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Insert\_Customers] Script Date: 07.12.2022 3:23:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER procedure [dbo].[Insert\_Customers]

@personalCode varchar (5),

@Ful\_name varchar(50),

@gender varchar(1),

@phone varchar(11)

as

insert into CUSTOMERS values (@personalCode,@Ful\_name,@gender,@phone)

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Update\_Customers] Script Date: 07.12.2022 3:23:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER proc [dbo].[Update\_Customers]

@id int,

@personalCode varchar (5),

@Ful\_name varchar(50),

@gender varchar(1),

@phone varchar(11)

as

begin

update CUSTOMERS

set Personal\_code= @personalCode, Full\_name = @Ful\_name, Gender = @gender, Phone\_number = @phone

where ID = @id

end

USE [Kursach]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Delete\_Place] Script Date: 07.12.2022 3:24:12 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER proc [dbo].[Delete\_Place]

@Id int

as begin

delete from PLACE

where ID = @Id

end

Код создания триггеров:

USE [Kursach]

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER trigger [dbo].[Control\_balance]

on [dbo].[MONETARY\_TRSNSACTION]

after delete

as

declare @balance money

declare @id int

select @balance = Transaction\_amount from deleted

select @id = Client\_ID from deleted

update CONTRACT

set Account\_balance = Account\_balance - @balance

where CUSTOMERS\_Number\_ID = @id

USE [Kursach]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[Data\_Control] Script Date: 07.12.2022 3:24:43 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER trigger [dbo].[Data\_Control]

on [dbo].[MONETARY\_TRSNSACTION]

after insert

as

begin

declare @date2 date

declare @date date

select @date2 = Date from inserted

set @date=Dateadd(day,1,CURRENT\_TIMESTAMP)

if(@date<@date2)

begin

raiserror ('Братик, ты что путешественник?',16,1)

rollback tran

end

USE [Kursach]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[Update\_balance] Script Date: 07.12.2022 3:25:09 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER trigger [dbo].[Update\_balance]

on [dbo].[MONETARY\_TRSNSACTION]

after insert, update

as

declare @balance money

declare @id int

select @balance = Transaction\_amount from inserted

select @id = Client\_ID from inserted

update CONTRACT

set Account\_balance = @balance + Account\_balance

where

CUSTOMERS\_Number\_ID = @id

ПРИЛОЖЕНИЕ Г

Код создания приложения.

Код Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач\_V2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

Program.user = textBox1.Text;

string pas = textBox2.Text;

if (Program.user == "Аналитик" && pas == "123")

{

Menu a = new Menu();

a.Show();

}

else if (Program.user == "Хелпер" && pas == "123")

{

Menu a = new Menu();

a.Show();

}

else if (Program.user == "Админ" && pas == "123")

{

Menu a = new Menu();

a.Show();

}

else

{

MessageBox.Show("Логин или пароль не верны! Попробуйте снова.", "Оповещение");

}

}

}

}

Код Menu:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач\_V2

{

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

Customers a = new Customers();

a.Show();

}

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

{

Contract a = new Contract();

a.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Monetary\_transaction a = new Monetary\_transaction();

a.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Place a = new Place();

a.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

Tariff a = new Tariff();

a.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

Statistic a = new Statistic();

a.Show();

}

private void Menu\_Load(object sender, EventArgs e)

{

if (Program.user == "Аналитик")

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

}

}

private void button7\_Click(object sender, EventArgs e)

{

MessageBox.Show("Работу выполнил Горшков Алексй. Группа ИП3-20", "Приложение к БД интерент-провайдера v2 ");

}

}

Код CONTRACT:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач\_V2

{

public partial class Contract : Form

{

public int peremennaya = 0;

public Contract()

{

InitializeComponent();

}

private void Contract\_Load(object sender, EventArgs e)

{

if (Program.user == "Хелпер")

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

}

LoadData();

}

private void LoadData()

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from Contract";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

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

{

try

{

Contract\_changes form = new Contract\_changes(((DataTable)dataGridView1.DataSource).NewRow(), false);

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

{

SqlConnection conn = new

SqlConnection(Курсач\_V2.Properties.Settings.Default.Connection);

SqlCommand com = new SqlCommand();

com.Connection = conn;

com.CommandType = CommandType.StoredProcedure;

com.CommandText = "Insert\_Contract";

com.Parameters.Add(new SqlParameter("@ID", SqlDbType.Int));

com.Parameters["@ID"].Value = form.Row["CUSTOMERS\_Number\_ID"];

com.Parameters.Add(new SqlParameter("@PersonalID", SqlDbType.Int));

com.Parameters["@PersonalID"].Value = form.Row["Personal\_account\_number"];

com.Parameters.Add(new SqlParameter("@TarifID", SqlDbType.Int));

com.Parameters["@TarifID"].Value = form.Row["TARIFF\_Tariff\_ID"];

com.Parameters.Add(new SqlParameter("@Datestare", SqlDbType.Date));

com.Parameters["@Datestare"].Value = form.Row["Date\_start"];

com.Parameters.Add(new SqlParameter("@Dateend", SqlDbType.Date));

com.Parameters["@Dateend"].Value = form.Row["Date\_end"];

com.Parameters.Add(new SqlParameter("@Balanse", SqlDbType.Money));

com.Parameters["@Balanse"].Value = form.Row["Account\_balance"];

com.Parameters.Add(new SqlParameter("@status", SqlDbType.VarChar));

com.Parameters["@status"].Value = form.Row["Contract\_status"];

com.Parameters.Add(new SqlParameter("@MoneyTransTuber", SqlDbType.Int));

com.Parameters["@MoneyTransTuber"].Value = form.Row["MONETARY\_TRSNSACTION\_Number\_ID"];

com.Parameters.Add(new SqlParameter("@PlaceNumber", SqlDbType.Int));

com.Parameters["@PlaceNumber"].Value = form.Row["PLACE\_Number\_ID"];

com.Parameters.Add(new SqlParameter("@Log", SqlDbType.VarChar));

com.Parameters["@Log"].Value = form.Row["Login"];

com.Parameters.Add(new SqlParameter("@Pas", SqlDbType.VarChar));

com.Parameters["@Pas"].Value = form.Row["Password"];

conn.Open();

if (com.ExecuteNonQuery() > 0)

{

MessageBox.Show("Контракт успешно добавлен.", "Оповещение");

}

conn.Close();

LoadData();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Selected = true;

}

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

catch (System.Data.SqlClient.SqlException)

{

MessageBox.Show("Ошибку внутренней логики приложния!", "Оповещение");

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

int id = (int)dataGridView1.SelectedRows[0].Cells["CUSTOMERS\_Number\_ID"].Value;

DataTable dt = (DataTable)dataGridView1.DataSource;

DataRow[] rows = dt.Select(String.Format("CUSTOMERS\_Number\_ID={0}", id));

Contract\_changes form = new Contract\_changes(rows[0], true);

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

{

SqlConnection conn = new

SqlConnection(Курсач\_V2.Properties.Settings.Default.Connection);

SqlCommand com = new SqlCommand();

com.Connection = conn;

com.CommandType = CommandType.StoredProcedure;

com.CommandText = "Update\_Contracts";

com.Parameters.Add(new SqlParameter("@PersonalID", SqlDbType.Int));

com.Parameters["@PersonalID"].Value = form.Row["Personal\_account\_number"];

com.Parameters.Add(new SqlParameter("@TarifID", SqlDbType.Int));

com.Parameters["@TarifID"].Value = form.Row["TARIFF\_Tariff\_ID"];

com.Parameters.Add(new SqlParameter("@Datestare", SqlDbType.Date));

com.Parameters["@Datestare"].Value = form.Row["Date\_start"];

com.Parameters.Add(new SqlParameter("@Dateend", SqlDbType.Date));

com.Parameters["@Dateend"].Value = form.Row["Date\_end"];

com.Parameters.Add(new SqlParameter("@Balanse", SqlDbType.Money));

com.Parameters["@Balanse"].Value = form.Row["Account\_balance"];

com.Parameters.Add(new SqlParameter("@status", SqlDbType.VarChar));

com.Parameters["@status"].Value = form.Row["Contract\_status"];

com.Parameters.Add(new SqlParameter("@MoneyTransTuber", SqlDbType.Int));

com.Parameters["@MoneyTransTuber"].Value = form.Row["MONETARY\_TRSNSACTION\_Number\_ID"];

com.Parameters.Add(new SqlParameter("@PlaceNumber", SqlDbType.Int));

com.Parameters["@PlaceNumber"].Value = form.Row["PLACE\_Number\_ID"];

com.Parameters.Add(new SqlParameter("@Log", SqlDbType.VarChar));

com.Parameters["@Log"].Value = form.Row["Login"];

com.Parameters.Add(new SqlParameter("@Pas", SqlDbType.VarChar));

com.Parameters["@Pas"].Value = form.Row["Password"];

conn.Open();

com.ExecuteNonQuery();

conn.Close();

LoadData();

MessageBox.Show("Изминения контракта внесены.", "Оповещение");

}

}

catch (System.ArgumentOutOfRangeException)

{

//MessageBox.Show("Вы не выбрали запись.", "Оповещение");

}

catch (System.ArgumentException)

{

//MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

int id = (int)dataGridView1.SelectedRows[0].Cells["CUSTOMERS\_Number\_ID"].Value;

if (MessageBox.Show("Удалить выделенный контракт?", "Удаление", MessageBoxButtons.OK) == DialogResult.OK)

{

SqlConnection conn = new SqlConnection(Курсач\_V2.Properties.Settings.Default.Connection);

SqlCommand com = new SqlCommand();

com.Connection = conn;

com.CommandType = CommandType.StoredProcedure;

com.CommandText = "Delete\_Contract";

com.Parameters.Add(new SqlParameter("@ID", SqlDbType.Int));

com.Parameters["@ID"].Value = id;

conn.Open();

com.ExecuteNonQuery();

conn.Close();

MessageBox.Show("Контракт успешно удален.", "Оповещение");

LoadData();

}

}

catch (System.ArgumentOutOfRangeException)

{

MessageBox.Show("Вы не выбрали запись.", "Оповещение");

}

}

private void button4\_Click(object sender, EventArgs e)

{

peremennaya = peremennaya + 1;

if (peremennaya % 2 == 0)

{

label1.Visible = false;

textBox1.Visible = false;

LoadData();

}

else

{

label1.Visible = true;

textBox1.Visible = true;

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from View\_6";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

SqlConnection conn = new SqlConnection(Курсач\_V2.Properties.Settings.Default.Connection);

SqlCommand com = new SqlCommand();

conn.Open();

com = new SqlCommand("Select \* from View\_6 where Full\_name like '%" + textBox1.Text + "%' ", conn);

DataTable dt = new DataTable();

dt.Load(com.ExecuteReader());

dataGridView1.DataSource = dt.DefaultView;

conn.Close();

}

}

}

Код EditCONTRACT:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач\_V2

{

public partial class Contract\_changes : Form

{

private DataRow row;

private bool isUpdate;

public Contract\_changes(DataRow inputRow, bool update)

{

InitializeComponent();

row = inputRow;

isUpdate = update;

}

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

{

row["CUSTOMERS\_Number\_ID"] = textBox1.Text;

row["Personal\_account\_number"] = textBox2.Text;

row["TARIFF\_Tariff\_ID"] = textBox3.Text;

row["Date\_start"] = maskedTextBox1.Text;

row["Date\_end"] = maskedTextBox2.Text;

row["Account\_balance"] = maskedTextBox3.Text;

row["Contract\_status"] = maskedTextBox4.Text;

row["MONETARY\_TRSNSACTION\_Number\_ID"] = maskedTextBox5.Text;

row["PLACE\_Number\_ID"] = maskedTextBox5.Text;

row["Login"] = textBox4.Text;

row["Password"] = textBox5.Text;

}

public DataRow Row

{

get { return row; }

set { row = value; }

}

private void Contract\_changes\_Load(object sender, EventArgs e)

{

if (isUpdate)

{

textBox1.Text = row["CUSTOMERS\_Number\_ID"].ToString();

textBox2.Text = row["Personal\_account\_number"].ToString();

textBox3.Text = row["TARIFF\_Tariff\_ID"].ToString();

maskedTextBox1.Text = row["Date\_start"].ToString();

maskedTextBox2.Text = row["Date\_end"].ToString();

maskedTextBox3.Text = row["Account\_balance"].ToString();

maskedTextBox4.Text = row["Contract\_status"].ToString();

maskedTextBox5.Text = row["MONETARY\_TRSNSACTION\_Number\_ID"].ToString();

maskedTextBox5.Text = row["PLACE\_Number\_ID"].ToString();

textBox4.Text = row["Login"].ToString();

textBox5.Text = row["Password"].ToString();

}

}

}

}

Код Statistic:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач\_V2

{

public partial class Statistic : Form

{

public Statistic()

{

InitializeComponent();

}

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

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from TarifCount";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from UsersInDB";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from UserAndTariff";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button4\_Click(object sender, EventArgs e)

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from UserContract";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button5\_Click(object sender, EventArgs e)

{

try

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = Properties.Settings.Default.Connection;

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = "Select \* from UserMoney";

conn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

for (int i = 0; i < rdr.FieldCount; i++)

{

dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));

}

while (rdr.Read())

{

DataRow row = dt.NewRow();

for (int i = 0; i < rdr.FieldCount; i++)

{

row[i] = rdr.GetValue(i);

}

dt.Rows.Add(row);

}

conn.Close();

dataGridView1.DataSource = dt;

}

catch (System.ArgumentException)

{

MessageBox.Show("Значение не соответствуют полям!", "Оповещение");

}

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Statistic\_Load(object sender, EventArgs e)

{

}

}

}