|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования и науки Российской Федерации  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **Московский государственный университет технологии и управления**  **имени К.Г. Разумовского (Первый казачий университет)**  **Университетский колледж информационных технологий** | | | |
| Специальность 09.02.03 Программирование в компьютерных системах | | | |
|  | | | |
| ПМ.02. Разработка и администрирование баз данных  МДК.02.02 Технология разработки и защиты баз данных | | | |
| Презентация на тему  *«Виды пользователей и привилегии»* | | | |
| Группа | | *П-303* |  |
| Студент | | *Личная подпись*  *(личная подпись)* | Мельдианов А.А |
| Преподаватель | | *Личная подпись*  *(личная подпись)* | Ларионова Е.А. |
|  |

**2016**

Оглавление

[1. Общая часть. 3](#_Toc454173443)

[1.1. Обзор и анализ предметной области. 3](#_Toc454173444)

[1.2. Объекты систем: 3](#_Toc454173445)

[1.3. Жизненный цикл базы данных 3](#_Toc454173446)

[1.4. Защита баз данных. 6](#_Toc454173447)

[1.5. Выбор и характеристика СУБД. 6](#_Toc454173448)

[1.6. Выбор и характеристика среды разработки приложения. 6](#_Toc454173449)

[2. Специальная часть. 7](#_Toc454173450)

[2.1. Постановка задачи. 7](#_Toc454173451)

[2.2. Архитектура информационной системы 7](#_Toc454173452)

[2.3. Логическая модель БД. 9](#_Toc454173453)

[2.4. Нормализация таблиц. 9](#_Toc454173454)

[2.5. Описание таблиц. 11](#_Toc454173455)

[2.6. Другие объекты базы данных (запросы, отчёты, представления, хранимые процедуры, триггеры) 12](#_Toc454173456)

[2.7. Разработка приложения. 14](#_Toc454173457)

[2.8. Инструкция пользователю. 15](#_Toc454173458)

[2.9. Настройка информационной системы. 15](#_Toc454173459)

[2.10. Формы ввода. 15](#_Toc454173460)

[Список литературы 19](#_Toc454173461)

[Приложение 1. Код программы 20](#_Toc454173462)

[Приложение 2. Sql скрипты 32](#_Toc454173463)

**Введение**

**Курсовой проект посвящен** разработке системы автоматизации продажи подержанных автомобилей.

**Целью курсового проекта является** – создание системы способной обеспечить более удобный контроль над менеджерами, а также для предоставления отчетности работы менеджеров.

Анализ существующих систем показал, что такой проект возможен. Во многих компаниях существуют системы для контроля работников и их рабочих процессов. Так же такие системы позволяют оперативно передавать необходимые данные для более высокого качества работы работников.

**Задачи проекта:**

* Анализ предметной области
* Проектирование базы данных средствами Erwin Modeler и MS SQL Server 2014
* Разработка приложения будет производиться с использованием языка программирования C# и IDE Visual studio 2015 и технологией WPF.
* Защита базы данных будет производиться с помощью разделения уровней доступа к базе данных и приложению.

**Возможности приложения**

* Подготовка к продаже авто
* Автоматизация продажи автомобилей
* Получение отчётов

## Общая часть.

### Обзор и анализ предметной области.

Курсовой проект посвящен разработки системы автоматизации продажи Б.У автомобилей. В нынешних реалиях существуют множество компаний по продаже Б\у автомобилей. Работа в такой компании всегда связанна с огромным количеством бумаг и работы с этими бумаги. Как бы было хорошо если бы что-то из огромной цепочки из документов стало более удобным и подстатье под нынешнее время.

При ведении статистики используется отчёты каждого менеджера продавшего автомобиля, так же при покупке машины компанией для будущей перепродажи.

Составление отчетов происходит раз в некоторое время так как работа то долгая. После их создания создаются отчеты по этим отчетам, и становится возможным вывести какую-либо статистику важную для компании. При таком подходе четко отслеживать работу менеджеров очень трудно из-за огромных задержек в виде составление отчётов.

В дополнение ко всему, не всегда есть возможность быстро узнать какие на самом деле есть у компании автомобили для продажи, а это может сыграть ключевую роль для компании и ее прибыли.

### Объекты систем:

Клиенты - Характеризуется ФИО, паспорт, пол.

Автомобили - Характеризуются Полной моделью автомобиля, его ценой, пройдённой дистанцией, дата производства.

Работники - Характеризуются ФИО, тип работника и Worker\_user – это специально идентификатор для входа в систему.

Сделки - Клиенты — это люди, купившие автомобиль у этой компании

### Жизненный цикл базы данных

Жизненным циклом базы данных называют процесс проектирования, реализации и управления БД.

1. Предварительное планирование:

* какие прикладные программы используются, и какие функции они выполняют;
* какие файлы связаны с каждым из этих приложений;
* какие новые приложения и файлы находятся в процессе работы.

Данная информация помогает определить, как используется информация

приложений, определить будущие требования к системе БД.

2. Проверка осуществимости:

* Технологическая осуществимость
  + Есть ли соответствующее программное обеспечение для реализации продукта
* Операционная осуществимость
  + Есть ли соответствующие разработчики, которые смогут принять участие в работе над проектом
* Экономическая целесообразность
  + Окупится ли проект, принесет ли прибыль
* Рациональность проекта:
  + В действительности ли конкретный проект подходит для осуществления поставленных целей.

3. Определение требований:

* Определяются цели системы путём анализа информационных потребностей. Здесь также обязательно указывается, какую именно БД следует создавать (распределённую, целостную) и какие коммуникационные средства необходимы. Выходной документ – комментарий, описывающий цели системы.
* Определение пользовательских требований: документация в виде обобщённой информации (комментарии, отчёты, опросы, анкеты и т. д.); фиксация функций системы и определение прикладных систем, которые будут выполнять эти требования. Данные представляются в виде соответствующих документов.
* Определение общих требований к оборудованию и программному обеспечению, связанных с поддержанием желаемого уровня быстродействия. (Выяснение количества пользователей системы, числа входных сообщений в день, количество распечаток). Данная информация используется для выбора типов компьютеров и СУБД, объёма дисков, количества принтеров. Данные этого этапа излагаются в отчёте, содержащем примерные конфигурации оборудования и программного обеспечения.
* Разработка плана поэтапного создания системы, включающий выбор исходных приложений.

4. Концептуальное проектирование:

* Создание концептуальной схемы БД. Спецификации разрабатываются в той степени, которая необходима для перехода к реализации.

5. Реализация:

1) Выбор и приобретение необходимой СУБД;

2) Преобразование концептуальной модели БД в логическую и физическую модель данных;

3) Заполнение базы данных;

4) Создание прикладных программ, контроль управления;

5) Обучение пользователей.

### Защита баз данных.

Методы защиты БД:

1. Защита паролем - представляет собой простой и эффективный способ защиты БД от несанкционированного доступа. Пароли устанавливаются пользователями или администраторами БД. Учет и хранение паролей выполняется самой СУБД. Обычно, пароли хранятся в определенных системных файлах СУБД в зашифрованном виде. После ввода пароля пользователю СУБД предоставляются все возможности по работе с БД.;
2. Шифрование данных и программ- это преобразование читаемого текста в нечитаемый текст, при помощи некоторого алгоритма; применяется для защиты уязвимых данных.;
3. Разграничение прав доступа к объектам БД - Права доступа определяют возможные действия над объектами. Владелец объекта (пользователь, создавший объект), а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа. Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.;
4. Защита полей и записей таблиц БД.

### Выбор и характеристика СУБД.

В качестве СУБД для проекта выступает Microsoft SQL Server 2014. Выбор обусловлен наличием опыта работы в данной СУБД и кодирования на языке SQL, а также удобством работы с ней в связке в выбранной IDE. Кроме того, СУБД существует в бесплатной редакции Express, а по студенческой подписке в сервисе Microsoft Azure можно бесплатно развернуть облачное хранилище на основе Microsoft SQL Server (Azure SQL). Из минусов СУБД следует отметить её низкую производительность на слабых машинах, а также отсутствие возможности работы на ОС, отличной от Windows.

### Выбор и характеристика среды разработки приложения.

В качестве среды разработки приложения выступает Microsoft Visual Studio 2015.Выбор обусловлен наличием опыта работы в данной IDE, а также желанием улучшить свои навыки в работе с ADO.NET. Выбранный язык разработки – C#.Среда разработки предназначена для работы в Windows, но существует кроссплатформенный текстовый редактор Visual Studio Code, который повторяет некоторую часть функционала Visual Studio. IDE обладает всем необходимым для разработки инструментарием, включая компилятор, отладчик, визуальный редактор и редактор кода с авто дополнением и всплывающими подсказками IntelliSense.IDE позволяет писать приложения для многих современных мобильных и настольных ОС, но ориентирована, в первую очередь, на Windows-разработчиков. Это и является одним из её главных минусов. Microsoft в настоящее время ведёт активную разработку кроссплатформенной версии .NET Framework, так что, остаётся надеяться, что в скором времени ситуация изменится.

Язык C# также способствует большой скорости разработки, поскольку лишён многих «опасных» вещей, таких как указатели, прямая работа с памятью. Для работы приложений, написанных на языке C# необходима установленная платформа .NET Framework.

## Специальная часть.

### Постановка задачи.

Задача предусматривает разработку ИС для осуществления продажи поддержаных автомобилей. Система должна предоставить рабочему персоналу удобный интерфейс, для работы с ИС. В ИС не должно быь возможности продать уже проданный автомобиль , Рабтник не должен видеть данные предназначеные для администраторов , а так же вход в ИС должены быть на основе логина и пароля созданныз при регистрации работника . Особых требований к интерфейсу приложения для администрирования не предъявляется.

### Архитектура информационной системы

Для создания ИС я использовал двухзвенную систему, как наиболее простую и удобную в реализации.

|  |  |
| --- | --- |
| Клиент  (Логика представления) | Сервер  (бизнес-логика, Хранение данных) |

Приложение

Microsoft SQL Server

Рисунок 1. Архитектура ИС

Двухзвенная архитектура характеризуется тем, что ИС разбивается на две части: клиент и сервер. На сервере происходит обработка бизнес-логики и там же осуществляется хранение данных, а клиент содержит лишь логику отображения данных, так называемый «тонкий клиент». Трёхзвенная архитектура отличается только тем, что разбивает сервер на две части: сервер бизнес-логики и сервер баз данных.

Применительно к данной информационной системе больше подходит двухзвенная модель, так как проект не содержит сложной бизнес-логики, которую следовало бы отделить от баз данных, не требует REST API или других интерфейсов, которые нельзя реализовать с помощью SQL Server.

### Логическая модель БД.

Для построения логической модели БД используется средства автоматизации проектирования. (Erwin data modeler)

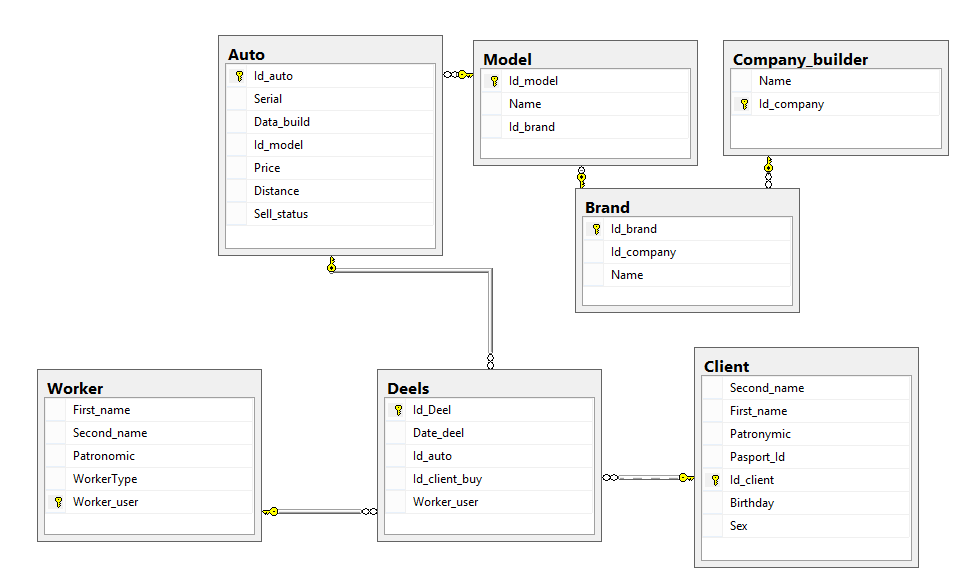


Рисунок 2. Логическая модель.

### Нормализация таблиц.

Сделать

1 нормальная форма БД приведена к 1 нормальной форме, т.к. все атрибуты просты, далее не делимы.

2 нормальна форма

3 нормальная форма Третья нормальная форма – если отношение находится во второй нормальной форме, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

Функциональные зависимости

|  |
| --- |
| First\_name |
| Second\_name |
| Patronomic |
| WorkerType |
| Worker\_user |

\*

|  |
| --- |
| Second\_name |
| First\_name |
| Patronomic |
| Pasport\_id |
| Id\_client |
| Birthday |
| sex |

\*

\*

|  |
| --- |
| Id\_deel |
| Date\_deel |
| Id\_auto |
| Id\_client\_buy |
| Worker\_user |

### Описание таблиц.

* + 1. **Worker** Таблица, отвечающая за хранение данных о работниках

|  |  |  |
| --- | --- | --- |
| First\_name | Varchar(20) | Имя |
| Second\_name | Varchar(20) | Фамилия |
| Patronomic | Varchar(50) | Отчество |
| WorkerType | Int | Тип работника |
| Worker\_user | Varchar(50) | Логин для присоединения к базе |

* + 1. **Client** Таблица, отекающая за хранение клиентов и их данных

|  |  |  |
| --- | --- | --- |
| Second\_name | Varchar(20) | Фамилия |
| First\_name | Varchar(20) | Имя |
| Patronomic | Varchar(50) | Отчество |
| Pasport\_id | Varchar(20) | Паспорт |
| Id\_client | Int | Униекальны идентификатор |
| Birthday | Date | Дата рождения |
| sex | bit | Пол(м\ж) |

* + 1. **Deels**

|  |  |  |
| --- | --- | --- |
| Id\_deel | Int | Идентификатор сделки |
| Date\_deel | Datetime | Дата сделки |
| Id\_auto | Int | Идентификатор автомобиля |
| Id\_client\_buy | Int | Идентификатор клиента |
| Worker\_user | Varchar(50) | Идентификатор работника |

* + 1. **Model**

|  |  |  |
| --- | --- | --- |
| Id\_model | Int | Идентификатор модели |
| Name | Varchar(20) | Название модели |
| Id\_brand | Int | Идентификатор бренда |

* + 1. **Brand**

|  |  |  |
| --- | --- | --- |
| Id\_brand | Int | Идентификатор бренда |
| Id\_company | Int | Идентификатор компании |
| Name | Varchar(20) | Название бренда |

* + 1. **Company**\_builder

|  |  |  |
| --- | --- | --- |
| Name | Varchar(20) | Название компании |
| Id\_company | Int | Идентификатор компании |

### Другие объекты базы данных (запросы, отчёты, представления, хранимые процедуры, триггеры)

* + 1. **Хранимы процедуры**

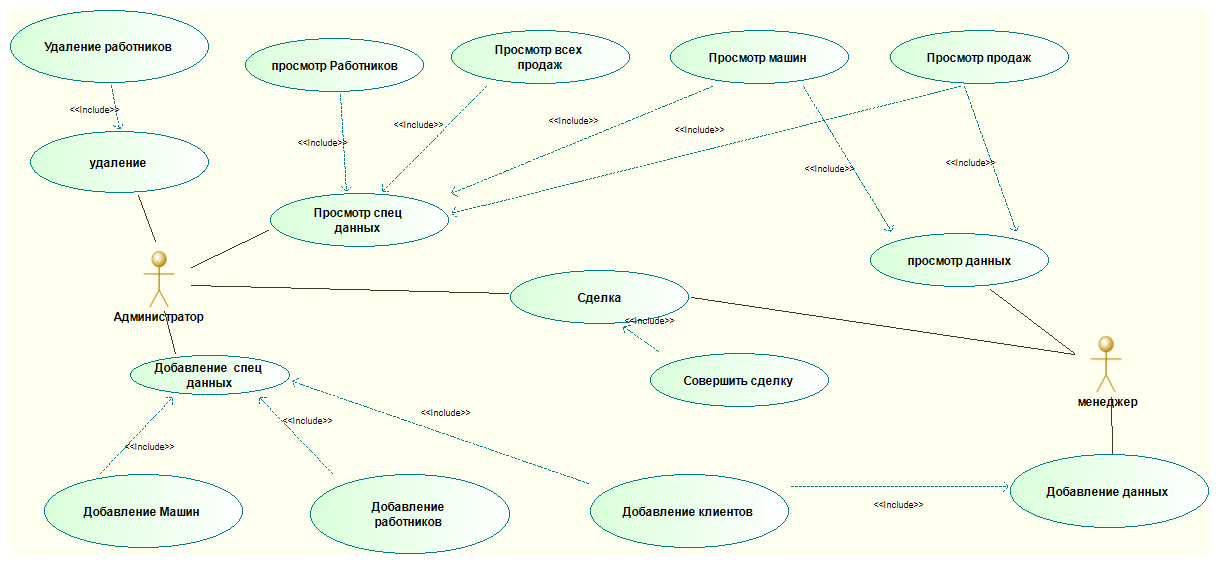
|  |  |
| --- | --- |
| Dbo.AddNewAuto | Добавление нового авто |
| Dbo.AddNewClient | Добавление нового клиента |
| Dbo.CallBackDataWorker | Возврат данных о работнике |
| Dbo.Chek\_Permissoin | Проверка прав |
| Dbo.CreateDeel | Создание сделки |
| Dbo.CreateUser | Создание нового работника |
| Dbo.DeleteUser | Удаление работника |
| Dbo.OpenViewDeels | Показ сделок работника |

* + 1. **Представления**

|  |  |
| --- | --- |
| Dbo.AutoInfoView | Вывод информации о машине |
| Dbo.ViewDeels | Вывод сделок |

### Разработка приложения.

#### Структурная схема приложения.



#### Используемая технология доступа к данным.

Для доступа к базе данных используется технология ADO.NET. Подключение происходит путем создания SqlConnection из пространства имен System.Data.SqlClient и создания SqlCommand для описания команды в Sql Server.

Пример хранимой процедуры См. Приложение.

#### Описание процесса отладки приложения.

Во время разработки приложения частые ошибки совершались из-за непонимания тех или иных пектов создания и рефакторинга баз данных, но это послужило большим толчком для изучения технологий SQL. примером может служить программа, найденная почти в конце разработки - **dbForge Studio for SQL Server.** Эта программа предоставляет удобные возможности для разработки и администрирования Баз данных. Также частые ошибки совершались из-за обращения к несуществующим хранимым процедурам или передача некорректных параметров.

#### Защита информационной системы.

В проекте использовалась система аутентификации на основе Sql-логина. Ограничения типов пользователей присутствуют не только в логике работы приложений, но также отражены в базе данных. Для каждого пользователя создаётся SQL Login и SQL Database User с SQL Role, соответствующей типу пользователя. У ролей есть права на исполнение только тех хранимых процедур, которые необходимы для соответствующего типа пользователя. Кроме того, прямого доступа к таблицам у пользователей нет.

«Manager\_admin» - это управляющий работниками, и он имеет право на создание и удаление работников, а также создание себе подобных, а также имеет все функции обычного работника.

Работник же имеет тип прав «Manager\_user» - он имеет право на чтение и запись данных в таблицы.

### Инструкция пользователю.

#### Общие сведения об информационной системе.

ИС предназначена работы работников компании, а также мониторинга и контроля их. Данная система позволит ускорить получение актуальных данных о каждом работнике компании и на основе его данных делать выводы.

#### Требования к техническим средствам.

Для работы ИС необходима следующая программная среда:

* Операционная система Microsoft Windows 7 или поздней версии;
* Установленная платформа .NET Framework 4.5
* Требования к программным средствам.
* Процессор – Int el, 2,5 Ггц или выше;
* Оперативная память не менее 2 Гб;
* Видеокарта и монитор;
* Клавиатура;
* Манипулятор "мышь";
* Сетевая карта Ethernet 100/1000 Мбит/с.

### Настройка информационной системы.

#### Настройка Sql Server

Для удобства разворота базы данных на машине был, разработав скрипт генератор, которые создаст всю структуру базы данных.

Файл скрипта называется «company generator script»

#### Настройка Основного приложения

Файл запуска приложения находится «\*\bin\release».

Имя пользователя по умолчанию «Manager\_admin – 123»

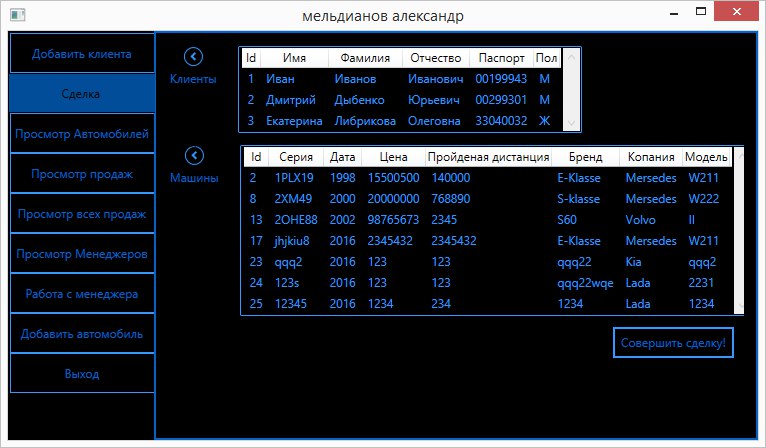
### Формы ввода.

Окно входа

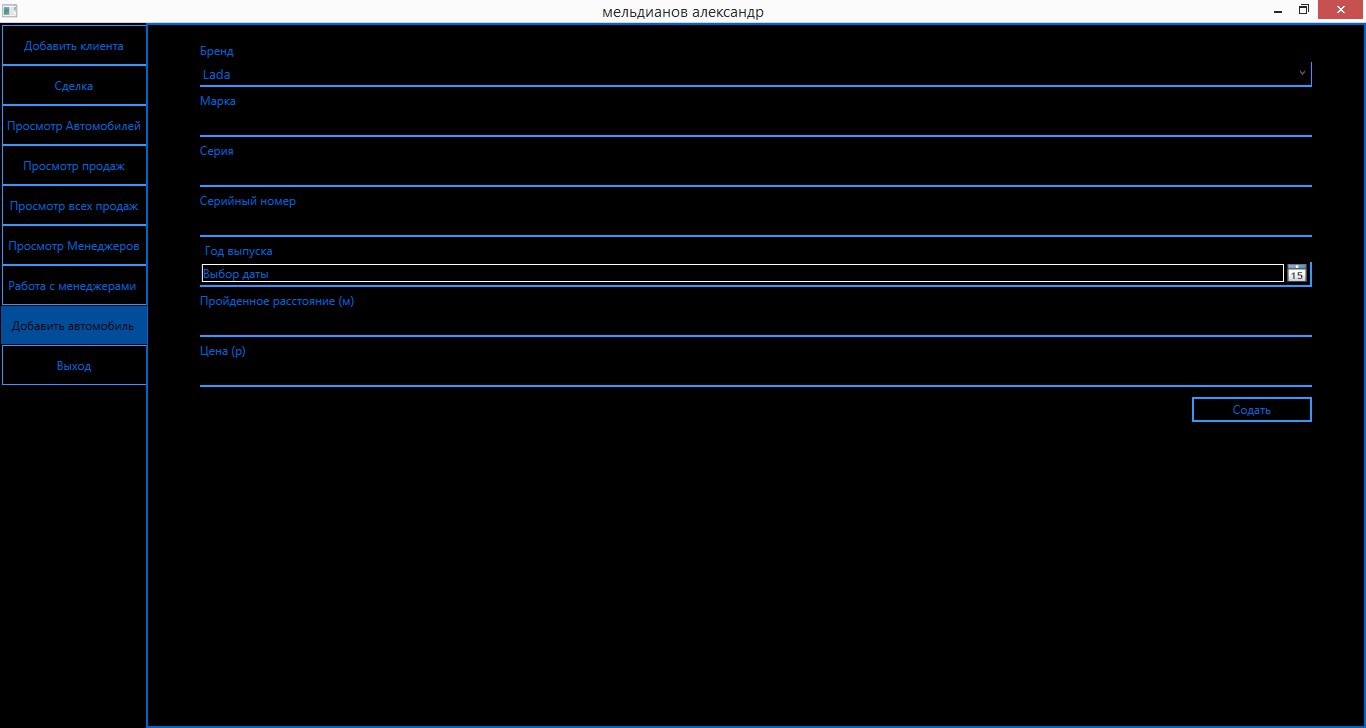
1 Окно входа



2 Окно сделки



3 Окно добавления Авто



4 Окно Создания работников



**Заключение**

Курсовой проект посвящен изучению методов разработки Системы для автоматизации продажи подержанных автомобилей, а также методы создания приложений для базы данных и добавления защиты к ней.

В процессе выполнения курсоого проекта были изучены способы и методы разратобки ИС с клиент-серверной архитектурой с помощью технологии ADO.NET ,языка программирования C# + WPF .

Изучена предметная область «продажа поддержаных автомобилей»

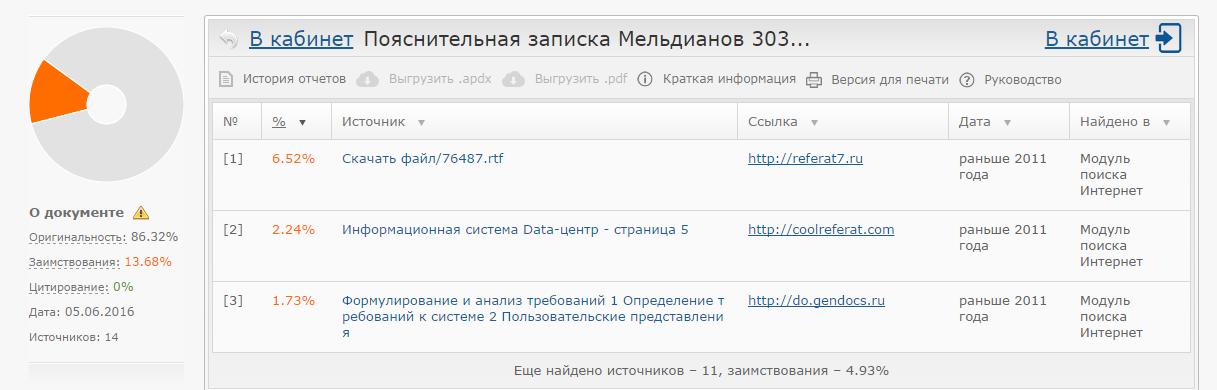
Спроектирована база данных для хранения и обработки необходимой информации.

Было разработано приложения для реализации работы с разработанной базой данных.

Была осуществлена защита посредством ADO.NET , Экранированием полей , и использованием хранимых процедур.

Были освоены методы создания и рефакторинга баз данных, связи приложения с помощью технологии ADO.NET

5 Анти плагиат



Репозитрий проекта (https://github.com/Neogara/Auto\_trade)

Дальнейшее развитие

* изменение интерфейса
* Расширение функционала приложения и базы данных

# Список литературы

* Сайт интуит - http://www.intuit.ru/studies/courses/5/5/info

# Приложение 1. Код программы

**WorkerCLass**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace AutoTrade

{

public class WorkerInfoClass

{

public string UserWorker { get; set; }

public string NameWorker { get; set; }

public string LastNameWorker { get; set; }

public string Patronomic { get; set; }

public int UserType { get; set; }

}

static public class WorkerClass

{

static public string UserWorker { get; set; }

static public string NameWorker { get; set; }

static public string LastNameWorker { get; set; }

static public string PatWorker { get; set; }

static public int UserType { get; set; }

static public string SqlConnect { get; set; }

static public string LoginIn (string login , string password)

{

SqlConnect = ConnectionStringBuilder.Build("SAMSUNG\\SQLEXPRESS", "Company", login, password);

var SqlServer = new SqlConnection(SqlConnect);

var cmd = new SqlCommand() {Connection = SqlServer };

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

string request;

if (login.ToUpper()=="Public\_user".ToUpper() & password=="123")

{

UserType = 0;

UserWorker = "publick\_user";

NameWorker = "Наблюдатель";

LastNameWorker = " ";

PatWorker = " ";

return "Done";

}

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.CommandText = "dbo.CallBackDataWorker";

adapter.SelectCommand = cmd;

try {

SqlServer.Open();

adapter.Fill(dataSet);

foreach (DataRow Row in dataSet.Tables[0].Rows)

{

NameWorker = Row[0].ToString();

LastNameWorker = Row[1].ToString();

PatWorker = Row[2].ToString();

UserType = (int)Row[3];

UserWorker = Row[4].ToString();

}

request = "Done";

}

catch (SqlException e )

{

request =string.Format( "Произошла ошибка \"{0}\" \n\n Код ошибки: {1} ",e.Message,e.Number);

}

finally

{

SqlServer.Close();

}

return request;

}

static public void LoginOut()

{

NameWorker = null;

UserType = 0;

LastNameWorker = null;

UserWorker = null;

SqlConnect = null;

}

}

}

**LoginWindow**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace AutoTrade

{

/// <summary>

/// Логика взаимодействия для Window1.xaml

/// </summary>

public partial class Window1 : Window

{

WorkWindows WorkPage = new WorkWindows();

public Window1()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

var request = WorkerClass.LoginIn(LoginText.Text, PasswordText.Password);

if (request != "Done")

{

MessageBox.Show(request);

return;

}

switch (WorkerClass.UserType)

{

case 1:

{

ViewTabItem(new TabItem[]{

WorkPage.AddClientItem,

WorkPage.AddDeelsItem,

WorkPage.ViewDeelsItem,

WorkPage.ViewAutoItem

});

break;

}

case 2:

{

ViewTabItem(new TabItem[] {

WorkPage.AddClientItem,

WorkPage.AddDeelsItem,

WorkPage.ViewAllDeelsItem,

WorkPage.ViewAllManagerItem,

WorkPage.ViewDeelsItem,

WorkPage.WorkWithManagerItem,

WorkPage.ViewAutoItem,

WorkPage.AddAuto

});

break;

}

default:

{

MessageBox.Show("Недопустимые права доступа ");

return;

}

}

WorkPage.Title = string.Format("{0} {1}", WorkerClass.LastNameWorker, WorkerClass.NameWorker);

WorkPage.Show();

this.Close();

}

private void ViewTabItem(TabItem[] ViewsTabitem)

{

WorkPage.AddClientItem.Visibility = Visibility.Collapsed;

WorkPage.AddDeelsItem.Visibility = Visibility.Collapsed;

WorkPage.ViewAllDeelsItem.Visibility = Visibility.Collapsed;

WorkPage.ViewAllManagerItem.Visibility = Visibility.Collapsed;

WorkPage.ViewDeelsItem.Visibility = Visibility.Collapsed;

WorkPage.WorkWithManagerItem.Visibility = Visibility.Collapsed;

WorkPage.ViewAutoItem.Visibility = Visibility.Collapsed;

WorkPage.AddAuto.Visibility = Visibility.Collapsed;

for (int i = 0; i <= ViewsTabitem.Length - 1; i++)

{

ViewsTabitem[i].Visibility = Visibility.Visible;

}

WorkPage.tabControl.SelectedValue = ViewsTabitem[0];

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

var request = WorkerClass.LoginIn("Public\_user", "123");

if (request == "Done")

{

ViewTabItem(new TabItem[] { WorkPage.ViewAutoItem });

WorkPage.Show();

this.Close();

}

else MessageBox.Show(request);

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

}

}

**WorkWindows**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Data;

namespace AutoTrade

{

public partial class WorkWindows : Window

{

public WorkWindows()

{

InitializeComponent();

}

private void button\_Click(object sender, RoutedEventArgs e)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var cmd = new SqlCommand() { Connection = SqlServer };

try

{

cmd.CommandText = "dbo.AddNewClient";

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("@Name", System.Data.SqlDbType.Char).Value = NameText.Text;

cmd.Parameters.Add("@Last\_name", System.Data.SqlDbType.Char).Value = Last\_nameText.Text;

cmd.Parameters.Add("@Patronymic", System.Data.SqlDbType.Char).Value = PatText.Text;

cmd.Parameters.Add("@Birthday", System.Data.SqlDbType.DateTime).Value = ClientDate.DisplayDate.Date;

cmd.Parameters.Add("@Sex", System.Data.SqlDbType.Bit).Value = SexComboBox.SelectedIndex;

cmd.Parameters.Add("@PasportId", System.Data.SqlDbType.Char).Value = PasportText.Text;

SqlServer.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("Клиент успешно добавлен");

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally

{ SqlServer.Close(); }

}

string ChekSex(bool Sex)

{

if (Sex) return "M";

else return "Ж";

}

void UpdateTablesOfDellWorker(ListView ListData)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "dbo.OpenViewDeels";

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("@Worker\_user", System.Data.SqlDbType.VarChar).Value = WorkerClass.UserWorker;

adapter.SelectCommand = cmd;

SqlServer.Open();

ListData.Items.Clear();

try

{

adapter.Fill(dataSet);

foreach (DataRow Row in dataSet.Tables[0].Rows)

{

var Table = new WorkerDeelsClass()

{

DateDeel = (DateTime)Row[1],

Serial = Row[2].ToString(),

Company = Row[3].ToString(),

Brand = Row[4].ToString(),

Model = Row[5].ToString(),

BuildDate = (DateTime)Row[6],

NameClient = Row[7].ToString(),

LastNameClient = Row[8].ToString(),

PasportId = Row[9].ToString(),

Price = (long)Row[10]

};

ListData.Items.Add(Table);

}

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

void UpdateTablesAllDels(ListView ListData)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "Select \* from dbo.ViewDeels";

cmd.CommandType = System.Data.CommandType.Text;

adapter.SelectCommand = cmd;

try

{

SqlServer.Open();

ListData.Items.Clear();

adapter.Fill(dataSet);

adapter.SelectCommand = cmd;

foreach (DataRow Row in dataSet.Tables[0].Rows)

{

var Table = new WorkerDeelsClass()

{

DateDeel = (DateTime)Row[1],

Serial = Row[2].ToString(),

Company = Row[3].ToString(),

Brand = Row[4].ToString(),

Model = Row[5].ToString(),

BuildDate = (DateTime)Row[6],

NameClient = Row[7].ToString(),

LastNameClient = Row[8].ToString(),

PasportId = Row[9].ToString(),

Price = (int)Row[10],

NameWorker = Row[12].ToString(),

LastNameWOrker = Row[13].ToString(),

Patronomic = Row[14].ToString()

};

ListData.Items.Add(Table);

}

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

void UpdateAutoTable(ListView ListData)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = " Select \* from dbo.AutoInfoView";

adapter.SelectCommand = cmd;

try

{

SqlServer.Open();

ListData.Items.Clear();

adapter.Fill(dataSet);

foreach (DataRow row in dataSet.Tables[0].Rows)

{

var auto = new AutoClass()

{

Serial = row[0].ToString(),

BuildDate = (DateTime)row[1],

Prise = (long)row[2],

Distanse = (int)row[3],

Brand = row[4].ToString(),

Company = row[5].ToString(),

Model = row[6].ToString(),

AutoId = row[8].ToString(),

};

ListData.Items.Add(auto);

}

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

void UpdateCLientTable()

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = " Select \* from dbo.Client";

adapter.SelectCommand = cmd;

try

{

SqlServer.Open();

dataGrid\_Copy1.Items.Clear();

adapter.Fill(dataSet);

foreach (DataRow row in dataSet.Tables[0].Rows)

{

var client = new ClientClass()

{

Name = row[0].ToString(),

LastName = row[1].ToString(),

Patronymic = row[2].ToString(),

Pasport = row[3].ToString(),

ClientId = row[4].ToString(),

Birthday = (DateTime)row[5],

Sex = ChekSex((bool)row[6]),

};

dataGrid\_Copy1.Items.Add(client);

}

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

void UpdateAllWorkers(ListView ListData)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var dataSet = new DataSet();

var adapter = new SqlDataAdapter();

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "Select \* from dbo.Worker";

cmd.CommandType = System.Data.CommandType.Text;

adapter.SelectCommand = cmd;

try

{

SqlServer.Open();

ListData.Items.Clear();

adapter.Fill(dataSet);

adapter.SelectCommand = cmd;

foreach (DataRow Row in dataSet.Tables[0].Rows)

{

var worker = new WorkerInfoClass()

{

NameWorker = Row[0].ToString(),

LastNameWorker = Row[1].ToString(),

Patronomic = Row[2].ToString(),

UserType = (int)Row[3],

UserWorker = Row[4].ToString(),

};

ListData.Items.Add(worker);

}

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

private void dataGrid\_Copy\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

expander.IsExpanded = false;

var item = dataGrid\_Copy.SelectedItem as AutoClass;

if (item != null)

{

AutoText.Text = string.Format("{0} {1} {2} , {3} года выпуска с пробегом {4} Km за {5} Рублей", item.Company, item.Brand, item.Model, item.BuildDate.Year, item.Distanse, item.Prise);

}

else

{

AutoText.Text = " ";

}

DeelClass.IdSelectAuto = int.Parse(item.AutoId);

}

private void TabItem\_Selected(object sender, RoutedEventArgs e)

{

UpdateAutoTable(dataGrid\_Copy);

UpdateCLientTable();

expander.IsExpanded = false;

expander\_Copy.IsExpanded = true;

}

private void dataGrid\_Copy1\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

expander.IsExpanded = true;

expander\_Copy.IsExpanded = false;

var item = dataGrid\_Copy1.SelectedItem as ClientClass;

if (item != null)

{

ClientText.Text = string.Format("{0} {1} {2} собирается купить ", item.LastName, item.Name, item.Patronymic);

}

else

{

ClientText.Text = " ";

}

DeelClass.IdSelectClient = int.Parse(item.ClientId);

}

private void CreateDeelButton\_Click(object sender, RoutedEventArgs e)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

if (DeelClass.IdSelectClient == 0)

{

MessageBox.Show("Неверно выбран Клиент");

return;

}

if (DeelClass.IdSelectAuto == 0)

{

MessageBox.Show("Неверно выбран Автомобиль");

return;

}

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "dbo.CreateDeel";

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("@Worker\_user", System.Data.SqlDbType.VarChar).Value = WorkerClass.UserWorker;

cmd.Parameters.Add("@id\_client", System.Data.SqlDbType.Int).Value = DeelClass.IdSelectClient;

cmd.Parameters.Add("@id\_auto", System.Data.SqlDbType.Int).Value = DeelClass.IdSelectAuto;

try

{

SqlServer.Open();

cmd.ExecuteNonQuery();

MessageBox.Show ("Сделка успешно совершена!");

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\" ", ex.Message));

}

SqlServer.Close();

UpdateAutoTable(dataGrid\_Copy);

UpdateCLientTable();

}

private void TabItem\_Selected\_1(object sender, RoutedEventArgs e)

{

UpdateAutoTable(dataGrid\_Copy3);

}

private void TabItem\_Selected\_2(object sender, RoutedEventArgs e)

{

UpdateTablesOfDellWorker(dataGrid\_Copy4);

}

private void TabItem\_Selected\_3(object sender, RoutedEventArgs e)

{

UpdateTablesAllDels(dataGrid\_Copy5);

}

private void butto2n\_Click(object sender, RoutedEventArgs e)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "dbo.CreateUser";

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("@Name", System.Data.SqlDbType.VarChar).Value = NameTex2t.Text;

cmd.Parameters.Add("@LastName", System.Data.SqlDbType.VarChar).Value = Last\_2nameText.Text;

cmd.Parameters.Add("@Pat", System.Data.SqlDbType.VarChar).Value = PatTe2xt.Text;

cmd.Parameters.Add("@Username", System.Data.SqlDbType.VarChar).Value = LoginText.Text;

cmd.Parameters.Add("@Password", System.Data.SqlDbType.VarChar).Value = PasswordText.Text;

cmd.Parameters.Add("@Usertype", System.Data.SqlDbType.Int).Value = UserTypes.SelectedIndex + 1;

try

{

SqlServer.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("Рабочий успешно добавлен");

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close();}

}

private void TextDistanse\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

}

}

private void ButtonAddAuto\_Click(object sender, RoutedEventArgs e)

{

var SqlServer = new SqlConnection(WorkerClass.SqlConnect);

var cmd = new SqlCommand() { Connection = SqlServer };

cmd.CommandText = "dbo.addNewAuto";

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("@Company", System.Data.SqlDbType.VarChar).Value = BrandBox.Text;

cmd.Parameters.Add("@Brand", System.Data.SqlDbType.VarChar).Value = TextMark.Text;

cmd.Parameters.Add("@Model", System.Data.SqlDbType.VarChar).Value = TextModel.Text;

cmd.Parameters.Add("@Serial", System.Data.SqlDbType.VarChar).Value = TextSerial.Text;

cmd.Parameters.Add("@Data", System.Data.SqlDbType.DateTime).Value = Date.DisplayDate.Date;

cmd.Parameters.Add("@Distanse", System.Data.SqlDbType.Int).Value = int.Parse(TextDistanse.Text);

cmd.Parameters.Add("@Price", System.Data.SqlDbType.BigInt).Value = Int64.Parse(TextPrice.Text);

try {

SqlServer.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("Авто добавлено!");

}

catch (Exception ex)

{

MessageBox.Show(string.Format("Произошла ошибка \"{0}\"", ex.Message));

}

finally { SqlServer.Close(); }

}

private void ViewAllManagerItem\_Selected(object sender, RoutedEventArgs e)

{

UpdateAllWorkers(dataGrid\_Copy6);

}

private void TabItem\_Selected\_4(object sender, RoutedEventArgs e)

{

WorkerClass.LoginOut();

Window1 LoginPage = new Window1();

LoginPage.Show();

this.Close();

}

}

}

# Приложение 2. Sql скрипты

USE master

GO

IF DB\_NAME() <> N'master' SET NOEXEC ON

-- Создать базу данных "Company"

PRINT (N'Создать базу данных "Company"')

GO

IF DB\_ID('Company') IS NULL

CREATE DATABASE Company

ON PRIMARY(

NAME = N'Company',

FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Company.mdf',

SIZE = 4288KB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1024KB

)

LOG ON(

NAME = N'Company\_log',

FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Company.ldf',

SIZE = 1024KB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 10%

)

GO

-- Изменить базу данных

PRINT (N'Изменить базу данных')

GO

ALTER DATABASE Company

SET

ANSI\_NULL\_DEFAULT ON,

ANSI\_NULLS ON,

ANSI\_PADDING ON,

ANSI\_WARNINGS ON,

ARITHABORT ON,

AUTO\_CLOSE OFF,

AUTO\_CREATE\_STATISTICS ON,

AUTO\_SHRINK OFF,

AUTO\_UPDATE\_STATISTICS ON,

AUTO\_UPDATE\_STATISTICS\_ASYNC OFF,

COMPATIBILITY\_LEVEL = 120,

CONCAT\_NULL\_YIELDS\_NULL ON,

CURSOR\_CLOSE\_ON\_COMMIT OFF,

CURSOR\_DEFAULT LOCAL,

DATE\_CORRELATION\_OPTIMIZATION OFF,

DB\_CHAINING OFF,

HONOR\_BROKER\_PRIORITY OFF,

MULTI\_USER,

NESTED\_TRIGGERS = ON,

NUMERIC\_ROUNDABORT OFF,

PAGE\_VERIFY CHECKSUM,

PARAMETERIZATION SIMPLE,

QUOTED\_IDENTIFIER ON,

READ\_COMMITTED\_SNAPSHOT OFF,

RECOVERY FULL,

RECURSIVE\_TRIGGERS OFF,

TRANSFORM\_NOISE\_WORDS = OFF,

TRUSTWORTHY OFF

WITH ROLLBACK IMMEDIATE

GO

ALTER DATABASE Company

COLLATE Cyrillic\_General\_CI\_AS

GO

ALTER DATABASE Company

SET DISABLE\_BROKER

GO

ALTER DATABASE Company

SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE Company

SET FILESTREAM (NON\_TRANSACTED\_ACCESS = OFF)

GO

--

-- Изменить владельца защищаемой сущности "Company" (база данных)

--

PRINT (N'Изменить владельца защищаемой сущности "Company" (база данных)')

GO

ALTER AUTHORIZATION ON DATABASE :: Company TO [Samsung\Ольга]

GO

--

-- Создать учетную запись "TestMan"

--

PRINT (N'Создать учетную запись "TestMan"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'TestMan')

CREATE LOGIN TestMan

WITH

PASSWORD = 0x0200E6FC25B652E991598D31B90FE49CAF096D5FE17F67D8E1E17937FCCA6B2B96937EAD74335E3F892721D4045E235BEFECDA13D9C4464AA8ACF57EB872688AC24CE9C17223 HASHED

,SID = 0xE8957D9B4355C64CBC9D0C64D122D8CF

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "TestAdm"

--

PRINT (N'Создать учетную запись "TestAdm"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'TestAdm')

CREATE LOGIN TestAdm

WITH

PASSWORD = 0x0200C3EEAC9C5AF7DB18A85BC025B47BE9A1D378105C1D8BB75BEB2DAD172ADF865CB032E5693C4BB74CB5C89BD734D75EEA04C9B4C39E158363C947AE04416AE96D2A79F51B HASHED

,SID = 0x1CDC2E26FE2A0949A2AE81C544270F24

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[Samsung\Ольга]"

--

PRINT (N'Создать учетную запись "[Samsung\Ольга]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'Samsung\Ольга')

CREATE LOGIN [Samsung\Ольга]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "Public\_user"

--

PRINT (N'Создать учетную запись "Public\_user"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'Public\_user')

CREATE LOGIN Public\_user

WITH

PASSWORD = 0x02001F490BF9B87730E9CD35FF5E1E730E22C4ADDE9224C3EE1BECEEBD6D0D977C50E164B3FA55F44789AA0BCBA34C7F8CC2C37DFCBFE5239E66F1E04054693CBF357E4736E7 HASHED

,SID = 0xDCB665E1CADCCC4997F6481D439D0041

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[NT SERVICE\Winmgmt]"

--

PRINT (N'Создать учетную запись "[NT SERVICE\Winmgmt]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'NT SERVICE\Winmgmt')

CREATE LOGIN [NT SERVICE\Winmgmt]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[NT SERVICE\SQLWriter]"

--

PRINT (N'Создать учетную запись "[NT SERVICE\SQLWriter]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'NT SERVICE\SQLWriter')

CREATE LOGIN [NT SERVICE\SQLWriter]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[NT SERVICE\ReportServer$SQLEXPRESS]"

--

PRINT (N'Создать учетную запись "[NT SERVICE\ReportServer$SQLEXPRESS]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'NT SERVICE\ReportServer$SQLEXPRESS')

CREATE LOGIN [NT SERVICE\ReportServer$SQLEXPRESS]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[NT Service\MSSQL$SQLEXPRESS]"

--

PRINT (N'Создать учетную запись "[NT Service\MSSQL$SQLEXPRESS]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'NT Service\MSSQL$SQLEXPRESS')

CREATE LOGIN [NT Service\MSSQL$SQLEXPRESS]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "[NT AUTHORITY\СИСТЕМА]"

--

PRINT (N'Создать учетную запись "[NT AUTHORITY\СИСТЕМА]"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'NT AUTHORITY\СИСТЕМА')

CREATE LOGIN [NT AUTHORITY\СИСТЕМА]

FROM WINDOWS

WITH

DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "Manager\_user"

--

PRINT (N'Создать учетную запись "Manager\_user"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'Manager\_user')

CREATE LOGIN Manager\_user

WITH

PASSWORD = 0x02000E4776BDA2371DFA68964CC0D5CBC9B218554C636A4E295A52BBE8516F6B9EEFBAEF4937C658813120E37B12CE5B18F2811220BD6161802B73459D7FDB3FB9624781A149 HASHED

,SID = 0x4888F707B2E26A4F91D0AAE587112F3A

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "Manager\_admin"

--

PRINT (N'Создать учетную запись "Manager\_admin"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'Manager\_admin')

CREATE LOGIN Manager\_admin

WITH

PASSWORD = 0x02005735409D78FAE2809A9B9866342D97A3ADFF7879C17017865FE790CD2B4336E492D30ABC6E3AF280D55D6E4EE061638FD1290537E9FFF63C7090C393CD3F4D43CE6DD0A5 HASHED

,SID = 0xA040222D4C04304E8CD02B7D3D9F6659

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "LoginovSI"

--

PRINT (N'Создать учетную запись "LoginovSI"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'LoginovSI')

CREATE LOGIN LoginovSI

WITH

PASSWORD = 0x0200A622A983A4D0488905B959D6E9DDC624FB8B1765FAF56450518E78C9D25DFB76AD22A77D59AF8ECF3B8E0623D0AB7353DB4692D8242619E33262D9B1727D58CF023962D1 HASHED

,SID = 0xDFB3C3F7E5BF254E987B97773269C57F

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "KirovaEN"

--

PRINT (N'Создать учетную запись "KirovaEN"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'KirovaEN')

CREATE LOGIN KirovaEN

WITH

PASSWORD = 0x02000D25894FFD08FD0FA087E06E1D688073067B9161D0E33111F7EE44E70CB083FA7557191BDA0F5A8BD8DB8E5D5B5CBE0E0CB56558E25DE62D3189F8C6FE8F8BD5ED162678 HASHED

,SID = 0xC214258302673E45A38F47632C8C4DEE

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "DibenkoDA"

--

PRINT (N'Создать учетную запись "DibenkoDA"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'DibenkoDA')

CREATE LOGIN DibenkoDA

WITH

PASSWORD = 0x0200A670E5430FC82FB3B734A3666E45BE03C80CDC70AC8E65D7B5DAFFC980F4108EADCD5D53E1B5081FBDC171866007A8583F247B7847F28BF31A0936D1E67457A8E51E2079 HASHED

,SID = 0x70DFA8D58B636B499F9B962773084E9F

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Создать учетную запись "aaa"

--

PRINT (N'Создать учетную запись "aaa"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'aaa')

CREATE LOGIN aaa

WITH

PASSWORD = 0x0200F36C8D4BE89852CF03928917EB8A307C1636D1E13E616434F1901A64C99BC20F4A7A408530D76004CC8DB523DD58A20036DFB9031A3355BC6F0ADBA556C582F62530AC44 HASHED

,SID = 0x04364C401674EE4BB5634E743DC14C63

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Изменить серверные разрешения для "aaa"

--

PRINT (N'Изменить серверные разрешения для "aaa"')

GO

GRANT CONNECT SQL TO aaa

GO

--

-- Создать учетную запись "##MS\_SQLResourceSigningCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_SQLResourceSigningCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_SQLResourceSigningCertificate##')

CREATE LOGIN ##MS\_SQLResourceSigningCertificate##

FROM CERTIFICATE ##MS\_SQLResourceSigningCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_SQLResourceSigningCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_SQLResourceSigningCertificate##"')

GO

GRANT VIEW ANY DEFINITION TO ##MS\_SQLResourceSigningCertificate##

GO

--

-- Создать учетную запись "##MS\_SQLReplicationSigningCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_SQLReplicationSigningCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_SQLReplicationSigningCertificate##')

CREATE LOGIN ##MS\_SQLReplicationSigningCertificate##

FROM CERTIFICATE ##MS\_SQLReplicationSigningCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_SQLReplicationSigningCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_SQLReplicationSigningCertificate##"')

GO

GRANT

AUTHENTICATE SERVER,

VIEW ANY DEFINITION,

VIEW SERVER STATE TO ##MS\_SQLReplicationSigningCertificate##

GO

--

-- Создать учетную запись "##MS\_SQLAuthenticatorCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_SQLAuthenticatorCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_SQLAuthenticatorCertificate##')

CREATE LOGIN ##MS\_SQLAuthenticatorCertificate##

FROM CERTIFICATE ##MS\_SQLAuthenticatorCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_SQLAuthenticatorCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_SQLAuthenticatorCertificate##"')

GO

GRANT AUTHENTICATE SERVER TO ##MS\_SQLAuthenticatorCertificate##

GO

--

-- Создать учетную запись "##MS\_SmoExtendedSigningCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_SmoExtendedSigningCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_SmoExtendedSigningCertificate##')

CREATE LOGIN ##MS\_SmoExtendedSigningCertificate##

FROM CERTIFICATE ##MS\_SmoExtendedSigningCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_SmoExtendedSigningCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_SmoExtendedSigningCertificate##"')

GO

GRANT VIEW ANY DEFINITION TO ##MS\_SmoExtendedSigningCertificate##

GO

--

-- Создать учетную запись "##MS\_PolicyTsqlExecutionLogin##"

--

PRINT (N'Создать учетную запись "##MS\_PolicyTsqlExecutionLogin##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_PolicyTsqlExecutionLogin##')

CREATE LOGIN ##MS\_PolicyTsqlExecutionLogin##

WITH

PASSWORD = 0x02005AFAB23F42B14BAD6767B3A76243D1A3C5DC321D141B1E189CCBD4690C5F98F464AADEC8D2C4882F3EE5D6F04B60A1FDF2A622C1319051DEF6E873346A5CBB30E72DBF9C HASHED

,SID = 0xA9EEB439F4762546BD90D163703F6DA1

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Изменить серверные разрешения для "##MS\_PolicyTsqlExecutionLogin##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_PolicyTsqlExecutionLogin##"')

GO

GRANT

CONNECT SQL,

VIEW ANY DEFINITION,

VIEW SERVER STATE TO ##MS\_PolicyTsqlExecutionLogin##

GO

--

-- Создать учетную запись "##MS\_PolicySigningCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_PolicySigningCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_PolicySigningCertificate##')

CREATE LOGIN ##MS\_PolicySigningCertificate##

FROM CERTIFICATE ##MS\_PolicySigningCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_PolicySigningCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_PolicySigningCertificate##"')

GO

GRANT

CONTROL SERVER,

VIEW ANY DEFINITION TO ##MS\_PolicySigningCertificate##

GO

--

-- Создать учетную запись "##MS\_PolicyEventProcessingLogin##"

--

PRINT (N'Создать учетную запись "##MS\_PolicyEventProcessingLogin##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_PolicyEventProcessingLogin##')

CREATE LOGIN ##MS\_PolicyEventProcessingLogin##

WITH

PASSWORD = 0x02000D4C78A4C82C6183F645E9C7B942630BC270B2F8867A108003CDA3E708A7DC0D0DFD56697E7584C15DCF377A769E89BA38E8B00B143248D9FC7668EDDD6B6E2443AD9C86 HASHED

,SID = 0x51D095ECDC6C164C9B6047F5838CAAA0

,DEFAULT\_DATABASE = master

,DEFAULT\_LANGUAGE = us\_english

GO

--

-- Изменить серверные разрешения для "##MS\_PolicyEventProcessingLogin##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_PolicyEventProcessingLogin##"')

GO

GRANT CONNECT SQL TO ##MS\_PolicyEventProcessingLogin##

GO

--

-- Создать учетную запись "##MS\_AgentSigningCertificate##"

--

PRINT (N'Создать учетную запись "##MS\_AgentSigningCertificate##"')

GO

IF NOT EXISTS (SELECT \* FROM master.dbo.syslogins WHERE loginname = N'##MS\_AgentSigningCertificate##')

CREATE LOGIN ##MS\_AgentSigningCertificate##

FROM CERTIFICATE ##MS\_AgentSigningCertificate##

GO

--

-- Изменить серверные разрешения для "##MS\_AgentSigningCertificate##"

--

PRINT (N'Изменить серверные разрешения для "##MS\_AgentSigningCertificate##"')

GO

GRANT CONNECT SQL TO ##MS\_AgentSigningCertificate##

GO

USE Company

GO

IF DB\_NAME() <> N'Company' SET NOEXEC ON

GO

--

-- Создать пользователя "TestMan"

--

PRINT (N'Создать пользователя "TestMan"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'TestMan') IS NULL

CREATE USER TestMan

WITHOUT LOGIN

GO

--

-- Создать пользователя "TestAdm"

--

PRINT (N'Создать пользователя "TestAdm"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'TestAdm') IS NULL

CREATE USER TestAdm

WITHOUT LOGIN

GO

--

-- Создать роль "Manager\_admin"

--

PRINT (N'Создать роль "Manager\_admin"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager\_admin') IS NULL

CREATE ROLE Manager\_admin

GO

--

-- Добавить члены в роль "Manager\_admin"

--

PRINT (N'Добавить члены в роль "Manager\_admin"')

GO

EXEC sp\_addrolemember N'Manager\_admin', N'TestAdm'

GO

--

-- Создать роль "Manager\_admin"

--

PRINT (N'Создать роль "Manager\_admin"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager\_admin') IS NULL

CREATE ROLE Manager\_admin

GO

--

-- Добавить члены в роль "Manager\_admin"

--

PRINT (N'Добавить члены в роль "Manager\_admin"')

GO

EXEC sp\_addrolemember N'Manager\_admin', N'TestAdm'

GO

--

-- Создать пользователя "Public\_user"

--

PRINT (N'Создать пользователя "Public\_user"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Public\_user') IS NULL

CREATE USER Public\_user

WITHOUT LOGIN

GO

--

-- Создать пользователя "NewTest12"

--

PRINT (N'Создать пользователя "NewTest12"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'NewTest12') IS NULL

CREATE USER NewTest12

WITHOUT LOGIN

GO

--

-- Создать пользователя "Madmin"

--

PRINT (N'Создать пользователя "Madmin"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Madmin') IS NULL

CREATE USER Madmin

WITHOUT LOGIN

GO

--

-- Создать пользователя "LoginovSI"

--

PRINT (N'Создать пользователя "LoginovSI"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'LoginovSI') IS NULL

CREATE USER LoginovSI

WITHOUT LOGIN

GO

--

-- Создать пользователя "KirovaEN"

--

PRINT (N'Создать пользователя "KirovaEN"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'KirovaEN') IS NULL

CREATE USER KirovaEN

WITHOUT LOGIN

GO

--

-- Создать пользователя "DibenkoDA"

--

PRINT (N'Создать пользователя "DibenkoDA"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'DibenkoDA') IS NULL

CREATE USER DibenkoDA

WITHOUT LOGIN

GO

--

-- Создать пользователя "aaa"

--

PRINT (N'Создать пользователя "aaa"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'aaa') IS NULL

CREATE USER aaa

WITHOUT LOGIN

GO

--

-- Создать таблицу "dbo.Worker"

--

PRINT (N'Создать таблицу "dbo.Worker"')

GO

IF OBJECT\_ID(N'dbo.Worker', 'U') IS NULL

CREATE TABLE dbo.Worker (

First\_name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Second\_name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Patronomic varchar(max) COLLATE Cyrillic\_General\_CI\_AS NULL,

WorkerType int NOT NULL DEFAULT (1),

Worker\_user varchar(50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT PK\_Worker PRIMARY KEY CLUSTERED (Worker\_user),

CONSTRAINT WorkerFirstNameCheck CHECK ([First\_name]<>''),

CONSTRAINT WorkerPatronomicCheck CHECK ([Patronomic]<>''),

CONSTRAINT WorkerSecondNameCheck CHECK ([Second\_name]<>'')

)

ON [PRIMARY]

TEXTIMAGE\_ON [PRIMARY]

GO

SET QUOTED\_IDENTIFIER, ANSI\_NULLS ON

GO

--

-- Создать процедуру "dbo.deleteUser"

--

GO

PRINT (N'Создать процедуру "dbo.deleteUser"')

GO

IF OBJECT\_ID(N'dbo.deleteUser', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.deleteUser

@login VARCHAR(50)

AS

DECLARE @Sql NVARCHAR(512)

DELETE FROM Worker

WHERE Worker\_user = @login

SET @Sql = ''drop LOGIN ['' + @login + '']''

EXEC (@Sql)

SET @Sql = ''drop user ['' + @login + '']''

EXEC (@Sql)

'

GO

--

-- Создать процедуру "dbo.CreateUser"

--

GO

PRINT (N'Создать процедуру "dbo.CreateUser"')

GO

IF OBJECT\_ID(N'dbo.CreateUser', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.CreateUser

@Username VARCHAR(128),

@Password VARCHAR(128),

@Usertype INT,

@Name VARCHAR(40),

@LastName VARCHAR(40),

@Pat VARCHAR(MAX)

AS

IF (NOT @Usertype IN (1, 2))

BEGIN

RAISERROR (''INVALID ARGUMENT @Usertype'', 16, 1)

RETURN 1

END

DECLARE @Sql NVARCHAR(512)

SET @Sql = ''CREATE LOGIN ['' + @Username + ''] WITH PASSWORD = '''''' + @Password + ''''''''

EXEC (@Sql)

SET @Sql = ''CREATE USER ['' + @Username + ''] FOR LOGIN ['' + @Username + '']''

EXEC (@Sql)

INSERT INTO Worker (First\_name, Second\_name, Patronomic, WorkerType, Worker\_user)

VALUES (@Name, @LastName, @Pat, CAST(@Usertype AS INT), @Username);

IF (@Usertype = 1)

BEGIN

EXEC sp\_addrolemember ''Manager\_user'',@Username

END

ELSE

IF (@Usertype = 2)

BEGIN

EXEC sp\_addrolemember N''Manager\_admin'',@Username

EXEC sp\_addrolemember N''db\_owner'' ,@Username

EXEC sp\_addrolemember N''db\_securityadmin '' ,@Username

EXEC sp\_addsrvrolemember @Username ,N''securityadmin''

END

'

GO

--

-- Создать таблицу "dbo.sysdiagrams"

--

PRINT (N'Создать таблицу "dbo.sysdiagrams"')

GO

IF OBJECT\_ID(N'dbo.sysdiagrams', 'U') IS NULL

CREATE TABLE dbo.sysdiagrams (

name sysname NOT NULL,

principal\_id int NOT NULL,

diagram\_id int IDENTITY,

version int NULL,

definition varbinary(max) NULL,

PRIMARY KEY CLUSTERED (diagram\_id),

CONSTRAINT UK\_principal\_name UNIQUE (principal\_id, name)

)

ON [PRIMARY]

TEXTIMAGE\_ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Model"

--

PRINT (N'Создать таблицу "dbo.Model"')

GO

IF OBJECT\_ID(N'dbo.Model', 'U') IS NULL

CREATE TABLE dbo.Model (

Id\_model int IDENTITY,

Name varchar(18) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Id\_brand int NOT NULL,

CONSTRAINT XPKModel PRIMARY KEY CLUSTERED (Id\_model),

CONSTRAINT UniqueId\_model UNIQUE (Id\_brand, Name),

UNIQUE (Name),

CONSTRAINT ModelNameCheck CHECK ([Name]<>'')

)

ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Deels"

--

PRINT (N'Создать таблицу "dbo.Deels"')

GO

IF OBJECT\_ID(N'dbo.Deels', 'U') IS NULL

CREATE TABLE dbo.Deels (

Id\_Deel int IDENTITY,

Date\_deel date NOT NULL,

Id\_auto int NOT NULL,

Id\_client\_buy int NOT NULL,

Worker\_user varchar(50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT XPKDeels PRIMARY KEY CLUSTERED (Id\_Deel)

)

ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Company\_builder"

--

PRINT (N'Создать таблицу "dbo.Company\_builder"')

GO

IF OBJECT\_ID(N'dbo.Company\_builder', 'U') IS NULL

CREATE TABLE dbo.Company\_builder (

Name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Id\_company int IDENTITY,

CONSTRAINT XPKCompany\_builder PRIMARY KEY CLUSTERED (Id\_company),

CONSTRAINT CompanyBuilderNameCheck CHECK ([Name]<>'')

)

ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Client"

--

PRINT (N'Создать таблицу "dbo.Client"')

GO

IF OBJECT\_ID(N'dbo.Client', 'U') IS NULL

CREATE TABLE dbo.Client (

Second\_name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

First\_name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Patronymic varchar(20) COLLATE Cyrillic\_General\_CI\_AS NULL,

Pasport\_Id varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Id\_client int IDENTITY,

Birthday date NOT NULL,

Sex bit NOT NULL,

CONSTRAINT XPKClient PRIMARY KEY CLUSTERED (Id\_client),

CONSTRAINT ClientFirstNameCheck CHECK ([First\_name]<>''),

CONSTRAINT ClientPasportIdCheck CHECK ([Pasport\_Id]<>''),

CONSTRAINT ClientPatronymicCheck CHECK ([Patronymic]<>''),

CONSTRAINT ClientSecondNameCheck CHECK ([Second\_name]<>'')

)

ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Brand"

--

PRINT (N'Создать таблицу "dbo.Brand"')

GO

IF OBJECT\_ID(N'dbo.Brand', 'U') IS NULL

CREATE TABLE dbo.Brand (

Id\_brand int IDENTITY,

Id\_company int NOT NULL,

Name varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT XPKBrand PRIMARY KEY CLUSTERED (Id\_brand),

CONSTRAINT BrandNameCheck CHECK ([Name]<>'')

)

ON [PRIMARY]

GO

--

-- Создать таблицу "dbo.Auto"

--

PRINT (N'Создать таблицу "dbo.Auto"')

GO

IF OBJECT\_ID(N'dbo.Auto', 'U') IS NULL

CREATE TABLE dbo.Auto (

Id\_auto int IDENTITY,

Serial varchar(20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

Data\_build date NOT NULL,

Id\_model int NOT NULL,

Price bigint NOT NULL,

Distance int NOT NULL,

Sell\_status bit NOT NULL,

CONSTRAINT XPKAuto000 PRIMARY KEY CLUSTERED (Id\_auto),

CONSTRAINT AutoDistanseChek CHECK ([Distance]>=(0)),

CONSTRAINT AutoPriceCheck CHECK ([Price]>=(0)),

CONSTRAINT AutoSerialChek CHECK ([Serial]<>'')

)

ON [PRIMARY]

GO

--

-- Создать процедуру "dbo.addNewAuto"

--

GO

PRINT (N'Создать процедуру "dbo.addNewAuto"')

GO

IF OBJECT\_ID(N'dbo.addNewAuto', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.addNewAuto

@Company VARCHAR(20),

@Brand VARCHAR(20),

@Model VARCHAR(20),

@Serial VARCHAR(20),

@Data datetime ,

@Distanse INT,

@Price BIGINT

AS

INSERT INTO Brand (Id\_company, Name)

VALUES ((SELECT cb.Id\_company FROM Company\_builder cb WHERE cb.Name = @Company) , @Brand);

INSERT INTO Model (Name, Id\_brand)

VALUES (@Model, (SELECT b.Id\_brand FROM Brand b WHERE b.Name = @Brand));

INSERT INTO Auto (Serial, Data\_build, Id\_model, Price, Distance, Sell\_status)

VALUES (@serial, @Data, (SELECT m.Id\_model FROM Model m WHERE m.Name = @Model), @Price, @Distanse, 0);

'

GO

--

-- Создать представление "dbo.ViewDeels"

--

GO

PRINT (N'Создать представление "dbo.ViewDeels"')

GO

IF OBJECT\_ID(N'dbo.ViewDeels', 'V') IS NULL

EXEC sp\_executesql N'CREATE VIEW dbo.ViewDeels

AS

SELECT dbo.Deels.Id\_Deel, dbo.Deels.Date\_deel, dbo.Auto.Serial, dbo.Company\_builder.Name, dbo.Brand.Name AS Expr1, dbo.Model.Name AS Expr2, dbo.Auto.Data\_build, dbo.Client.Second\_name,

dbo.Client.First\_name, dbo.Client.Pasport\_Id, dbo.Auto.Price, dbo.Deels.Worker\_user, dbo.Worker.First\_name AS Expr3, dbo.Worker.Second\_name AS Expr4, dbo.Worker.Patronomic

FROM dbo.Deels INNER JOIN

dbo.Worker ON dbo.Deels.Worker\_user = dbo.Worker.Worker\_user INNER JOIN

dbo.Auto ON dbo.Deels.Id\_auto = dbo.Auto.Id\_auto INNER JOIN

dbo.Client ON dbo.Deels.Id\_client\_buy = dbo.Client.Id\_client INNER JOIN

dbo.Model ON dbo.Auto.Id\_model = dbo.Model.Id\_model INNER JOIN

dbo.Brand ON dbo.Model.Id\_brand = dbo.Brand.Id\_brand INNER JOIN

dbo.Company\_builder ON dbo.Brand.Id\_company = dbo.Company\_builder.Id\_company

'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPane1" для "dbo.ViewDeels" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPane1" для "dbo.ViewDeels" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPane1', 'SCHEMA', N'dbo', 'TABLE', N'ViewDeels', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPane1', N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[41] 4[29] 2[22] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Deels"

Begin Extent =

Top = 6

Left = 38

Bottom = 136

Right = 208

End

DisplayFlags = 280

TopColumn = 1

End

Begin Table = "Worker"

Begin Extent =

Top = 6

Left = 246

Bottom = 136

Right = 416

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Auto"

Begin Extent =

Top = 6

Left = 454

Bottom = 136

Right = 624

End

DisplayFlags = 280

TopColumn = 3

End

Begin Table = "Client"

Begin Extent =

Top = 138

Left = 38

Bottom = 268

Right = 208

End

DisplayFlags = 280

TopColumn = 3

End

Begin Table = "Model"

Begin Extent =

Top = 138

Left = 454

Bottom = 251

Right = 624

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Brand"

Begin Extent =

Top = 208

Left = 222

Bottom = 321

Right = 392

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Company\_builder"

Begin Extent =

Top = 121

Left = 262

Bottom = 217

Right = 432

End

DisplayFlags = 280

', 'SCHEMA', N'dbo', 'VIEW', N'ViewDeels'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPane2" для "dbo.ViewDeels" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPane2" для "dbo.ViewDeels" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPane2', 'SCHEMA', N'dbo', 'TABLE', N'ViewDeels', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPane2', N' TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 11

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

End

End

', 'SCHEMA', N'dbo', 'VIEW', N'ViewDeels'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPaneCount" для "dbo.ViewDeels" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPaneCount" для "dbo.ViewDeels" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPaneCount', 'SCHEMA', N'dbo', 'TABLE', N'ViewDeels', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPaneCount', 2, 'SCHEMA', N'dbo', 'VIEW', N'ViewDeels'

GO

--

-- Создать представление "dbo.AutoInfoView"

--

GO

PRINT (N'Создать представление "dbo.AutoInfoView"')

GO

IF OBJECT\_ID(N'dbo.AutoInfoView', 'V') IS NULL

EXEC sp\_executesql N'CREATE VIEW dbo.AutoInfoView

AS

SELECT dbo.Auto.Serial, dbo.Auto.Data\_build AS [Data\_build], dbo.Auto.Price, dbo.Auto.Distance AS Distanse, dbo.Brand.Name AS Name\_Brand, dbo.Company\_builder.Name AS Name\_company,

dbo.Model.Name AS Name\_model, dbo.Auto.Id\_model, dbo.Auto.Id\_auto

FROM dbo.Company\_builder INNER JOIN

dbo.Brand ON dbo.Company\_builder.Id\_company = dbo.Brand.Id\_company INNER JOIN

dbo.Model ON dbo.Brand.Id\_brand = dbo.Model.Id\_brand INNER JOIN

dbo.Auto ON dbo.Model.Id\_model = dbo.Auto.Id\_model

WHERE (dbo.Auto.Sell\_status = 0)

'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPane1" для "dbo.AutoInfoView" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPane1" для "dbo.AutoInfoView" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPane1', 'SCHEMA', N'dbo', 'TABLE', N'AutoInfoView', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPane1', N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[41] 4[27] 2[14] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Company\_builder"

Begin Extent =

Top = 6

Left = 454

Bottom = 102

Right = 624

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Brand"

Begin Extent =

Top = 6

Left = 246

Bottom = 119

Right = 416

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Model"

Begin Extent =

Top = 102

Left = 454

Bottom = 215

Right = 624

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Auto"

Begin Extent =

Top = 6

Left = 38

Bottom = 136

Right = 208

End

DisplayFlags = 280

TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 9

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

', 'SCHEMA', N'dbo', 'VIEW', N'AutoInfoView'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPane2" для "dbo.AutoInfoView" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPane2" для "dbo.AutoInfoView" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPane2', 'SCHEMA', N'dbo', 'TABLE', N'AutoInfoView', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPane2', N' End

End

', 'SCHEMA', N'dbo', 'VIEW', N'AutoInfoView'

GO

--

-- Добавить расширенное свойство "MS\_DiagramPaneCount" для "dbo.AutoInfoView" (представление)

--

PRINT (N'Добавить расширенное свойство "MS\_DiagramPaneCount" для "dbo.AutoInfoView" (представление)')

GO

IF NOT EXISTS (SELECT 1 FROM sys.fn\_listextendedproperty(N'MS\_DiagramPaneCount', 'SCHEMA', N'dbo', 'TABLE', N'AutoInfoView', NULL, NULL))

EXEC sys.sp\_addextendedproperty N'MS\_DiagramPaneCount', 2, 'SCHEMA', N'dbo', 'VIEW', N'AutoInfoView'

GO

--

-- Создать роль приложения "Manager"

--

PRINT (N'Создать роль приложения "Manager"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager') IS NULL

CREATE APPLICATION ROLE Manager

WITH PASSWORD = 'passw0rd'

GO

--

-- Создать роль "Manager\_user"

--

PRINT (N'Создать роль "Manager\_user"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NULL

CREATE ROLE Manager\_user

GO

--

-- Добавить члены в роль "Manager\_user"

--

PRINT (N'Добавить члены в роль "Manager\_user"')

GO

EXEC sp\_addrolemember N'Manager\_user', N'aaa'

GO

EXEC sp\_addrolemember N'Manager\_user', N'DibenkoDA'

GO

EXEC sp\_addrolemember N'Manager\_user', N'KirovaEN'

GO

EXEC sp\_addrolemember N'Manager\_user', N'LoginovSI'

GO

EXEC sp\_addrolemember N'Manager\_user', N'Manager'

GO

EXEC sp\_addrolemember N'Manager\_user', N'NewTest12'

GO

EXEC sp\_addrolemember N'Manager\_user', N'Public\_user'

GO

EXEC sp\_addrolemember N'Manager\_user', N'TestMan'

GO

--

-- Создать роль "Manager\_user"

--

PRINT (N'Создать роль "Manager\_user"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NULL

CREATE ROLE Manager\_user

GO

--

-- Добавить члены в роль "Manager\_user"

--

PRINT (N'Добавить члены в роль "Manager\_user"')

GO

EXEC sp\_addrolemember N'Manager\_user', N'aaa'

GO

EXEC sp\_addrolemember N'Manager\_user', N'DibenkoDA'

GO

EXEC sp\_addrolemember N'Manager\_user', N'KirovaEN'

GO

EXEC sp\_addrolemember N'Manager\_user', N'LoginovSI'

GO

EXEC sp\_addrolemember N'Manager\_user', N'Manager'

GO

EXEC sp\_addrolemember N'Manager\_user', N'NewTest12'

GO

EXEC sp\_addrolemember N'Manager\_user', N'Public\_user'

GO

EXEC sp\_addrolemember N'Manager\_user', N'TestMan'

GO

--

-- Создать процедуру "dbo.OpenViewDeels"

--

GO

PRINT (N'Создать процедуру "dbo.OpenViewDeels"')

GO

IF OBJECT\_ID(N'dbo.OpenViewDeels', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.OpenViewDeels

(

@Worker\_user VARCHAR(50)

)

AS

select \* from dbo.ViewDeels where ViewDeels.Worker\_user = @Worker\_user

'

GO

--

-- Предоставить разрешения на "dbo.OpenViewDeels"

--

PRINT (N'Предоставить разрешения на "dbo.OpenViewDeels"')

GO

IF OBJECT\_ID(N'dbo.OpenViewDeels', 'P') IS NOT NULL

AND DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NOT NULL

GRANT EXECUTE ON dbo.OpenViewDeels TO Manager\_user

GO

--

-- Создать процедуру "dbo.CreateDeel"

--

GO

PRINT (N'Создать процедуру "dbo.CreateDeel"')

GO

IF OBJECT\_ID(N'dbo.CreateDeel', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.CreateDeel

(

@Worker\_user VARCHAR(50) ,

@id\_client int ,

@id\_auto int

)

AS

BEGIN

if (select Sell\_status from Auto where Id\_auto = @id\_auto) = 0

begin

insert into dbo.Deels( Date\_deel,Id\_auto , Id\_client\_buy,Worker\_user )

values (CURRENT\_TIMESTAMP, @id\_auto,@id\_client,@Worker\_user)

update Auto

set auto.Sell\_status = 1

where Id\_auto = @id\_auto

end

else THROW 50001, ''Автомобиль уже продан '', 1;

END

'

GO

--

-- Предоставить разрешения на "dbo.CreateDeel"

--

PRINT (N'Предоставить разрешения на "dbo.CreateDeel"')

GO

IF OBJECT\_ID(N'dbo.CreateDeel', 'P') IS NOT NULL

AND DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NOT NULL

GRANT EXECUTE ON dbo.CreateDeel TO Manager\_user

GO

--

-- Создать процедуру "dbo.Chek\_Permisson"

--

GO

PRINT (N'Создать процедуру "dbo.Chek\_Permisson"')

GO

IF OBJECT\_ID(N'dbo.Chek\_Permisson', 'P') IS NULL

EXEC sp\_executesql N'

CREATE PROCEDURE dbo.Chek\_Permisson

@Login NVARCHAR(256)

as

DECLARE @user NVARCHAR(256);

EXECUTE AS LOGIN = @login;

SELECT @login = login\_name FROM sys.dm\_exec\_sessions WHERE session\_id = @@SPID;

SELECT @user = d.name

FROM sys.database\_principals AS d

INNER JOIN sys.server\_principals AS s

ON d.sid = s.sid

WHERE s.name = @login;

SELECT u.name, r.name

FROM sys.database\_role\_members AS m

INNER JOIN sys.database\_principals AS r

ON m.role\_principal\_id = r.principal\_id

INNER JOIN sys.database\_principals AS u

ON u.principal\_id = m.member\_principal\_id

WHERE u.name = @user;

--SELECT class\_desc, major\_id, permission\_name, state\_desc

-- FROM sys.database\_permissions

-- WHERE grantee\_principal\_id = USER\_ID(@user);

REVERT;'

GO

--

-- Предоставить разрешения на "dbo.Chek\_Permisson"

--

PRINT (N'Предоставить разрешения на "dbo.Chek\_Permisson"')

GO

IF OBJECT\_ID(N'dbo.Chek\_Permisson', 'P') IS NOT NULL

AND DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NOT NULL

GRANT EXECUTE ON dbo.Chek\_Permisson TO Manager\_user

GO

--

-- Создать процедуру "dbo.CallBackDataWorker"

--

GO

PRINT (N'Создать процедуру "dbo.CallBackDataWorker"')

GO

IF OBJECT\_ID(N'dbo.CallBackDataWorker', 'P') IS NULL

EXEC sp\_executesql N'CREATE PROCEDURE dbo.CallBackDataWorker

AS

SELECT \* FROM Worker WHERE Worker\_user= (SELECT CURRENT\_USER )

'

GO

--

-- Предоставить разрешения на "dbo.CallBackDataWorker"

--

PRINT (N'Предоставить разрешения на "dbo.CallBackDataWorker"')

GO

IF OBJECT\_ID(N'dbo.CallBackDataWorker', 'P') IS NOT NULL

AND DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NOT NULL

GRANT EXECUTE ON dbo.CallBackDataWorker TO Manager\_user

GO

--

-- Создать процедуру "dbo.AddNewClient"

--

GO

PRINT (N'Создать процедуру "dbo.AddNewClient"')

GO

IF OBJECT\_ID(N'dbo.AddNewClient', 'P') IS NULL

EXEC sp\_executesql N'

Create PROCEDURE dbo.AddNewClient

(

@Name char(20) ,

@Last\_name char(20) ,

@Patronymic char(20) ,

@Birthday datetime ,

@Sex Bit ,

@PasportId char (20)

)

AS

BEGIN

insert into dbo.Client (First\_name,Second\_name,Patronymic,Birthday,sex,Pasport\_Id)

values (@name , @Last\_name ,@Patronymic , @Birthday,@sex,@PasportId)

END'

GO

--

-- Предоставить разрешения на "dbo.AddNewClient"

--

PRINT (N'Предоставить разрешения на "dbo.AddNewClient"')

GO

IF OBJECT\_ID(N'dbo.AddNewClient', 'P') IS NOT NULL

AND DATABASE\_PRINCIPAL\_ID(N'Manager\_user') IS NOT NULL

GRANT EXECUTE ON dbo.AddNewClient TO Manager\_user

GO

--

-- Создать функцию "dbo.fn\_diagramobjects"

--

GO

PRINT (N'Создать функцию "dbo.fn\_diagramobjects"')

GO

IF NOT EXISTS (SELECT 1 FROM sys.objects WHERE object\_id = OBJECT\_ID(N'dbo.fn\_diagramobjects') AND type IN ('IF', 'FN', 'TF'))

EXEC sp\_executesql N'

CREATE FUNCTION dbo.fn\_diagramobjects()

RETURNS int

WITH EXECUTE AS N''dbo''

AS

BEGIN

declare @id\_upgraddiagrams int

declare @id\_sysdiagrams int

declare @id\_helpdiagrams int

declare @id\_helpdiagramdefinition int

declare @id\_creatediagram int

declare @id\_renamediagram int

declare @id\_alterdiagram int

declare @id\_dropdiagram int

declare @InstalledObjects int

select @InstalledObjects = 0

select @id\_upgraddiagrams = object\_id(N''dbo.sp\_upgraddiagrams''),

@id\_sysdiagrams = object\_id(N''dbo.sysdiagrams''),

@id\_helpdiagrams = object\_id(N''dbo.sp\_helpdiagrams''),

@id\_helpdiagramdefinition = object\_id(N''dbo.sp\_helpdiagramdefinition''),

@id\_creatediagram = object\_id(N''dbo.sp\_creatediagram''),

@id\_renamediagram = object\_id(N''dbo.sp\_renamediagram''),

@id\_alterdiagram = object\_id(N''dbo.sp\_alterdiagram''),

@id\_dropdiagram = object\_id(N''dbo.sp\_dropdiagram'')

if @id\_upgraddiagrams is not null

select @InstalledObjects = @InstalledObjects + 1

if @id\_sysdiagrams is not null

select @InstalledObjects = @InstalledObjects + 2

if @id\_helpdiagrams is not null

select @InstalledObjects = @InstalledObjects + 4

if @id\_helpdiagramdefinition is not null

select @InstalledObjects = @InstalledObjects + 8

if @id\_creatediagram is not null

select @InstalledObjects = @InstalledObjects + 16

if @id\_renamediagram is not null

select @InstalledObjects = @InstalledObjects + 32

if @id\_alterdiagram is not null

select @InstalledObjects = @InstalledObjects + 64

if @id\_dropdiagram is not null

select @InstalledObjects = @InstalledObjects + 128

return @InstalledObjects

END

'

GO

--

-- Предоставить разрешения на "dbo.fn\_diagramobjects"

--

PRINT (N'Предоставить разрешения на "dbo.fn\_diagramobjects"')

GO

IF EXISTS (SELECT 1 FROM sys.objects WHERE object\_id = OBJECT\_ID(N'dbo.fn\_diagramobjects') AND type IN ('IF', 'FN', 'TF'))

AND DATABASE\_PRINCIPAL\_ID(N'guest') IS NOT NULL

DENY EXECUTE ON dbo.fn\_diagramobjects TO guest

GO

IF EXISTS (SELECT 1 FROM sys.objects WHERE object\_id = OBJECT\_ID(N'dbo.fn\_diagramobjects') AND type IN ('IF', 'FN', 'TF'))

AND DATABASE\_PRINCIPAL\_ID(N'public') IS NOT NULL

GRANT EXECUTE ON dbo.fn\_diagramobjects TO [public]

GO

--

-- Создать роль приложения "Manager"

--

PRINT (N'Создать роль приложения "Manager"')

GO

IF DATABASE\_PRINCIPAL\_ID(N'Manager') IS NULL

CREATE APPLICATION ROLE Manager

WITH PASSWORD = 'passw0rd'

GO

--

-- Вывод данных для таблицы Auto

--

SET IDENTITY\_INSERT dbo.Auto ON

GO

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (2, N'1PLX19', '1998-01-01', 3, 15500500, 140000, CONVERT(bit, 'True'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (8, N'2XM49', '2000-01-01', 2, 20000000, 768890, CONVERT(bit, 'False'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (12, N'2KKO54', '2015-01-01', 5, 30049332, 234554, CONVERT(bit, 'True'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (13, N'2OHE88', '2002-01-01', 6, 98765673, 2345, CONVERT(bit, 'False'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (17, N'jhjkiu8', '2016-05-10', 3, 2345432, 2345432, CONVERT(bit, 'False'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (23, N'qqq2', '2016-05-11', 21, 123, 123, CONVERT(bit, 'False'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (24, N'123s', '2016-05-11', 22, 123, 123, CONVERT(bit, 'False'))

INSERT dbo.Auto(Id\_auto, Serial, Data\_build, Id\_model, Price, Distance, Sell\_status) VALUES (25, N'12345', '2016-05-22', 23, 1234, 234, CONVERT(bit, 'True'))

GO

SET IDENTITY\_INSERT dbo.Auto OFF

GO

--

-- Вывод данных для таблицы Brand

--

SET IDENTITY\_INSERT dbo.Brand ON

GO

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (1, 1, N'S-klasse')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (2, 1, N'E-Klasse')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (4, 2, N'XC90')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (5, 2, N'S60')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (6, 3, N'Astra')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (7, 3, N'Zafira')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (8, 4, N'X6')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (9, 4, N'X5')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (10, 4, N'qqq')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (11, 5, N'qqq')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (12, 5, N'qqq')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (13, 5, N'qqq')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (14, 6, N'qqq')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (15, 6, N'qqq22')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (16, 5, N'qqq22wqe')

INSERT dbo.Brand(Id\_brand, Id\_company, Name) VALUES (17, 5, N'1234')

GO

SET IDENTITY\_INSERT dbo.Brand OFF

GO

--

-- Вывод данных для таблицы Client

--

SET IDENTITY\_INSERT dbo.Client ON

GO

INSERT dbo.Client(Second\_name, First\_name, Patronymic, Pasport\_Id, Id\_client, Birthday, Sex) VALUES (N'Иван', N'Иванов', N'Иванович', N'00199943', 1, '1960-01-04', CONVERT(bit, 'True'))

INSERT dbo.Client(Second\_name, First\_name, Patronymic, Pasport\_Id, Id\_client, Birthday, Sex) VALUES (N'Дмитрий', N'Дыбенко', N'Юрьевич', N'00299301', 2, '1970-03-05', CONVERT(bit, 'True'))

INSERT dbo.Client(Second\_name, First\_name, Patronymic, Pasport\_Id, Id\_client, Birthday, Sex) VALUES (N'Екатерина', N'Либрикова', N'Олеговна', N'33040032', 3, '1990-05-21', CONVERT(bit, 'False'))

GO

SET IDENTITY\_INSERT dbo.Client OFF

GO

--

-- Вывод данных для таблицы Company\_builder

--

SET IDENTITY\_INSERT dbo.Company\_builder ON

GO

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Mersedes', 1)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Volvo', 2)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Opel', 3)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'BMW', 4)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Lada', 5)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Kia', 6)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Hyundai', 7)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Renault', 8)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Toyota', 9)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Nissan', 10)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Volkswagen', 11)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'Skoda', 13)

INSERT dbo.Company\_builder(Name, Id\_company) VALUES (N'CHevrolet', 14)

GO

SET IDENTITY\_INSERT dbo.Company\_builder OFF

GO

--

-- Вывод данных для таблицы Deels

--

SET IDENTITY\_INSERT dbo.Deels ON

GO

INSERT dbo.Deels(Id\_Deel, Date\_deel, Id\_auto, Id\_client\_buy, Worker\_user) VALUES (1005, '2016-05-29', 2, 1, N'TestAdm')

INSERT dbo.Deels(Id\_Deel, Date\_deel, Id\_auto, Id\_client\_buy, Worker\_user) VALUES (1006, '2016-05-30', 25, 1, N'TestAdm')

GO

SET IDENTITY\_INSERT dbo.Deels OFF

GO

--

-- Вывод данных для таблицы Model

--

SET IDENTITY\_INSERT dbo.Model ON

GO

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (2, N'W222', 1)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (3, N'W211', 2)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (5, N'I', 4)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (6, N'II', 5)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (7, N'J', 6)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (8, N'B', 7)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (9, N'E71', 8)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (15, N'E70', 9)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (16, N'qqq', 10)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (21, N'qqq2', 15)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (22, N'2231', 16)

INSERT dbo.Model(Id\_model, Name, Id\_brand) VALUES (23, N'1234', 17)

GO

SET IDENTITY\_INSERT dbo.Model OFF

GO

--

-- Вывод данных для таблицы sysdiagrams

--

GO

SET IDENTITY\_INSERT dbo.sysdiagrams OFF

GO

--

-- Вывод данных для таблицы Worker

--

INSERT dbo.Worker(First\_name, Second\_name, Patronomic, WorkerType, Worker\_user) VALUES (N'2', N'1', N'3', 1, N'aaa')

INSERT dbo.Worker(First\_name, Second\_name, Patronomic, WorkerType, Worker\_user) VALUES (N'Дмитрий', N'Дыбенко', N'Алексеевич', 1, N'DibenkoDA')

INSERT dbo.Worker(First\_name, Second\_name, Patronomic, WorkerType, Worker\_user) VALUES (N'Елена', N'Кирова', N'Николаевна', 1, N'KirovaEN')

INSERT dbo.Worker(First\_name, Second\_name, Patronomic, WorkerType, Worker\_user) VALUES (N'Сергей', N'Логинов', N'Иванович', 1, N'LoginovSI')

INSERT dbo.Worker(First\_name, Second\_name, Patronomic, WorkerType, Worker\_user) VALUES (N'александр', N'мельдианов', N'алексеевич', 2, N'TestAdm')

GO

USE Company

GO

IF DB\_NAME() <> N'Company' SET NOEXEC ON

GO

--

-- Создать внешний ключ "R\_37" для объекта типа таблица "dbo.Brand"

--

PRINT (N'Создать внешний ключ "R\_37" для объекта типа таблица "dbo.Brand"')

GO

IF OBJECT\_ID('dbo.R\_37', 'F') IS NULL

AND OBJECT\_ID('dbo.Brand', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Company\_builder', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Brand', N'Id\_company') IS NOT NULL

AND COL\_LENGTH(N'dbo.Company\_builder', N'Id\_company') IS NOT NULL

ALTER TABLE dbo.Brand

ADD CONSTRAINT R\_37 FOREIGN KEY (Id\_company) REFERENCES dbo.Company\_builder (Id\_company) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Создать внешний ключ "R\_38" для объекта типа таблица "dbo.Model"

--

PRINT (N'Создать внешний ключ "R\_38" для объекта типа таблица "dbo.Model"')

GO

IF OBJECT\_ID('dbo.R\_38', 'F') IS NULL

AND OBJECT\_ID('dbo.Model', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Brand', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Model', N'Id\_brand') IS NOT NULL

AND COL\_LENGTH(N'dbo.Brand', N'Id\_brand') IS NOT NULL

ALTER TABLE dbo.Model

ADD CONSTRAINT R\_38 FOREIGN KEY (Id\_brand) REFERENCES dbo.Brand (Id\_brand) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Создать внешний ключ "R\_39" для объекта типа таблица "dbo.Auto"

--

PRINT (N'Создать внешний ключ "R\_39" для объекта типа таблица "dbo.Auto"')

GO

IF OBJECT\_ID('dbo.R\_39', 'F') IS NULL

AND OBJECT\_ID('dbo.Auto', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Model', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Auto', N'Id\_model') IS NOT NULL

AND COL\_LENGTH(N'dbo.Model', N'Id\_model') IS NOT NULL

ALTER TABLE dbo.Auto

ADD CONSTRAINT R\_39 FOREIGN KEY (Id\_model) REFERENCES dbo.Model (Id\_model) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Создать внешний ключ "FK\_Deels\_Worker\_Worker\_user" для объекта типа таблица "dbo.Deels"

--

PRINT (N'Создать внешний ключ "FK\_Deels\_Worker\_Worker\_user" для объекта типа таблица "dbo.Deels"')

GO

IF OBJECT\_ID('dbo.FK\_Deels\_Worker\_Worker\_user', 'F') IS NULL

AND OBJECT\_ID('dbo.Deels', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Worker', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Deels', N'Worker\_user') IS NOT NULL

AND COL\_LENGTH(N'dbo.Worker', N'Worker\_user') IS NOT NULL

ALTER TABLE dbo.Deels

ADD CONSTRAINT FK\_Deels\_Worker\_Worker\_user FOREIGN KEY (Worker\_user) REFERENCES dbo.Worker (Worker\_user) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Создать внешний ключ "R\_29" для объекта типа таблица "dbo.Deels"

--

PRINT (N'Создать внешний ключ "R\_29" для объекта типа таблица "dbo.Deels"')

GO

IF OBJECT\_ID('dbo.R\_29', 'F') IS NULL

AND OBJECT\_ID('dbo.Deels', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Auto', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Deels', N'Id\_auto') IS NOT NULL

AND COL\_LENGTH(N'dbo.Auto', N'Id\_auto') IS NOT NULL

ALTER TABLE dbo.Deels

ADD CONSTRAINT R\_29 FOREIGN KEY (Id\_auto) REFERENCES dbo.Auto (Id\_auto) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Создать внешний ключ "R\_42" для объекта типа таблица "dbo.Deels"

--

PRINT (N'Создать внешний ключ "R\_42" для объекта типа таблица "dbo.Deels"')

GO

IF OBJECT\_ID('dbo.R\_42', 'F') IS NULL

AND OBJECT\_ID('dbo.Deels', 'U') IS NOT NULL

AND OBJECT\_ID('dbo.Client', 'U') IS NOT NULL

AND COL\_LENGTH(N'dbo.Deels', N'Id\_client\_buy') IS NOT NULL

AND COL\_LENGTH(N'dbo.Client', N'Id\_client') IS NOT NULL

ALTER TABLE dbo.Deels WITH NOCHECK

ADD CONSTRAINT R\_42 FOREIGN KEY (Id\_client\_buy) REFERENCES dbo.Client (Id\_client) ON DELETE CASCADE ON UPDATE CASCADE

GO

--

-- Выключить внешний ключ "R\_42" для объекта типа таблица "dbo.Deels"

--

PRINT (N'Выключить внешний ключ "R\_42" для объекта типа таблица "dbo.Deels"')

GO

ALTER TABLE dbo.Deels

NOCHECK CONSTRAINT R\_42

GO

SET NOEXEC OFF

Go