МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА—Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
(наименование института, филиала)
Кафедра КБ-3 «Безопасность программных решений»
(наименование кафедры)

Утверждаю
Заведующий кафедрой КБ-3:
Горин Д.С. _____
(Ф.И.О.) (подпись)
«12» февраля 2022 г.

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине Принципы, технологии и средства организации данных компонентов и программного обеспечения
(наименование дисциплины)

Тема курсовой работы Создание клиент-серверного приложения для работы с базой данных

Студент Банщиков Михаил Михайлович БСБО-01-20 _____
(учебная группа, фамилия, имя, отчество студента) (подпись студента)

Исходные данные Вариант № 16

Перечень вопросов, подлежащих обработке, и обязательного графического материала:
Создание отлаженного клиентского приложения для работы с БД
Создание БД, схемы таблиц, индексов, хранимых функций, процедур, триггеров, транзакций
Реализация прав доступа к объектам БД, подключение к БД и демонстрация работы с учетом различий по доступу в программе-клиенте
Реализация в программе-клиенте запросов согласно заданию

Срок предоставление к защите курсовой работы до «19» мая 2022 г.

Задание на курсовую работу выдал Филатов В.В. _____
(Ф.И.О. руководителя) (подпись руководителя)
«12» 02 2022г.

Задание на курсовую работу получил Банщиков М. М. _____
(Ф.И.О. исполнителя) (подпись исполнителя

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА—Российский технологический университет»

РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

(наименование института, филиала)

Кафедра КБ-3 «Безопасность программных решений»

(наименование кафедры)

КУРСОВАЯ РАБОТА

по дисциплине Принципы, технологии и средства организации данных компонентов и программного обеспечения

(наименование дисциплины)

Тема курсовой работы Создание клиент-серверного приложения для работы с базой данной

Студент группы__БСБО-01-20_Бандурин Михаил Михайлович __

учебная группа, фамилия, имя, отчество студента подпись студента

Руководитель курсовой работы доцент, к.т.н. ____ /Филатов В.В./

должность, звание, ученая ~~степень,~~ подпись руководителя

Рецензент (при наличии)_____

должность, звание, ученая степень подпись рецензента

Работа представлена к защите «_____»_ _20__г.

Допущен к защите «_____»_ _20_

Москва 2022

Оглавление

Задание

Разработать клиент-серверное приложение, серверная часть которого должна быть реализована на Microsoft SQL Server или PostgreSQL, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области реализовать заданную (по варианту) схему отношений, т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме.

16	Электронный кадастр объектов недвижимости и земельных участков предприятия	Ведение справочников и учет объектов недвижимости и земельных участков, на которых они расположены (на одном участке может расположено несколько объектов). Информация о владельцах земли и недвижимости и т.п.	3

Вариант 3

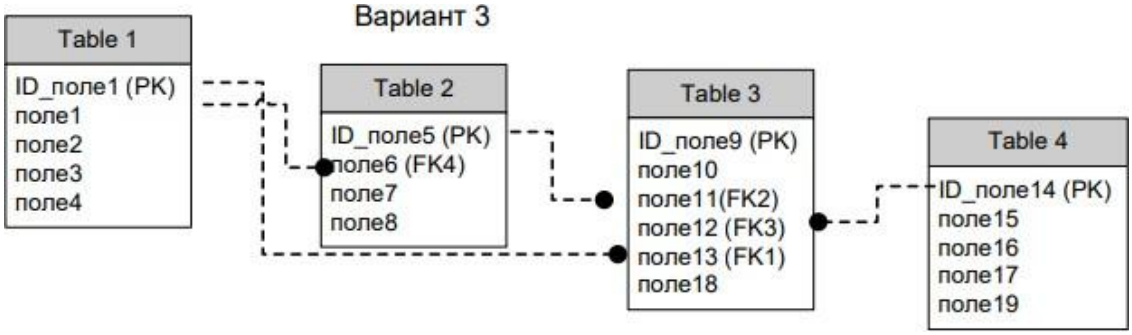
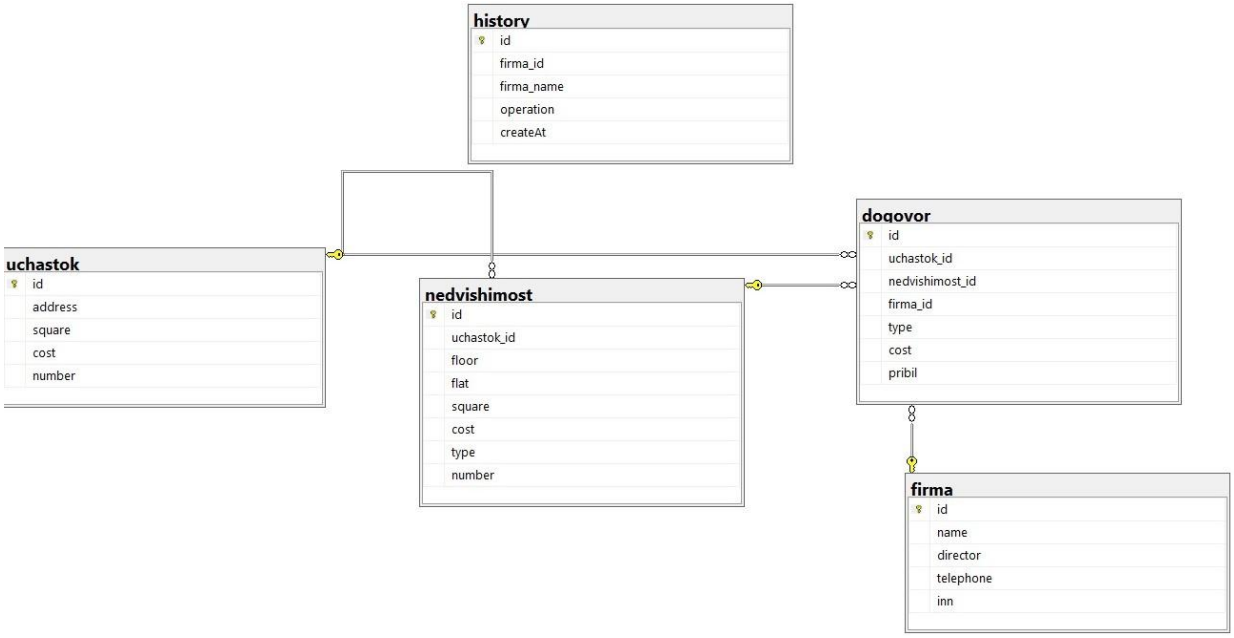


Table 1
ID_поле1 (PK)
поле1
поле2
поле3
поле4
Table 2
ID_поле5 (PK)
поле6 (FK4)
поле7
поле8
Table 3
ID_поле9 (PK)
поле10
поле11(FK2)
поле12 (FK3)
поле13 (FK1)
поле18
Table 4
ID_поле14 (PK)
поле15
поле16
поле17
поле19

Схема БД


history
id
firma_id
firma_name
operation
createAt
dogovor
id
uchastok_id
nedvishimost_id
firma_id
type
cost
pribil
uchastok
id
address
square
cost
number
nedvishimost
id
uchastok_id
floor
flat
square
cost
type
number
firma
id
name
director
telephone
inn

Введение

В данной курсовой работе задачей было разработать клиенто-серверное приложение для электронного реестра. Для решения этой задачи как сервер использовался Microsoft SQL Server, а для клиентского интерфейса – язык программирования c#. Для соединения этих частей использовалась библиотека Microsoft Data SQL Client. Для разделения прав доступа в базе данных с целью защиты информации, использовалось разделение на две роли: администратора и пользовательской. “Администратор” имеет полные права на базу данных, “юзер” – может получать только выборки и менять значения в представлении view. Сама база данных представляет из себя 5 таблиц, включающих в себя: таблицу с информацией о участках, недвижимости, договорах, и фирмах покупателях историю операций над таблицей, содержащей в себе фирмы; триггеров и курсоров; связей между таблицами, с помощью внешних ключей, процедуры, позволяющие заполнять, изменять и удалять данные из таблицы, процедуры и функции, возвращающие табличные и скалярные значения; индексы.

Внешние ключи соединяют: участки и договора, недвижимость и договора, фирмы и договора, участки и недвижимость; передавая значения идентификаторов записей в первых таблицах во вторые.

В клиентском интерфейсе предусмотрена защита от неправильно введённых данных. При несоответствии внешнего ключа с идентификаторами той таблице, которую он соединяет, будет введена ошибка и запрос не будет отправлен в БД, также присутствует защита от ввода неверного типа данных.

Техническая информация

В данной главе разобрана работа приложения со стороны клиента. При запуске приложения пользователь попадает в окно авторизации. В введении было указано, что в данной БД предусмотрено два пользователя, для каждого из них имеется пароль. После успешной авторизации клиент попадает в главное меню. На нём находятся кнопки для перехода к выборкам, к основным четырём таблицам (представление и таблица с историей операций находятся в отдельном меню).

Форма для работы с основными таблицами динамическая, при нажатии на кнопку, создаётся конструктор формы с параметрами, которые прописаны в функции нажатия кнопки.

В форме для основных таблиц находятся: таблица, сделанная при помощи элемента Windows Form– DataGridView и три кнопки button, предназначенные для добавления, обновления и удаления данных. При загрузке формы происходит передача данных из параметра в переменные класса формы, затем столбцам из этих данных присваивается имя, а также с помощью запроса SELECT в БД таблица заполняется данными, кроме идентификатора, который заполняет отдельный массив. При удалении данных из таблицы извлекается индекс выделенной строки, который используется в качестве индекса массива с идентификатором, затем создаётся и отправляется запрос в БД. Для заполнения полей для добавления и изменения данных существет отдельная форма, вызываемая в виде диалога.

Также в этот диалог попадают типы полей и ограничения. После нажатия на кнопку “Добавить”/”Изменить”, поля проходят проверку на соответствия с указанными типами данных при помощи команды Try.Parse, в случае несоответствия выдаётся ошибка, не дающая закрыть диалоговое окно параметром “OK”, также данные подстраиваются под формат SQL команды. При удачном завершении диалога из него получают обработанные данные,

которые затем добавляют в строку с текстом запроса. При изменении данных, в начале поля диалогового окна заполнено исходными данными.

Существует ещё два меню для отдельных запросов, функций, процедур и курсора. В одном меню находятся, не требующие от пользователя ввода аргументов, в другом требующие. С помощью данных меню, запускаются функции.

База данных

Создание таблиц

При создании таблиц использовались такие типы данных, как: NVARCHAR- для текстовых данных, INT - для целочисленных данных, FLOAT – для дробных данных. Также использовались первичные и внешние ключи, ограничения уникальности – UNIQUE. В таблице историй операции использовалось самозаполняющееся поле с системной датой и временем.

```
CREATE TABLE uchastok
(
id INT IDENTITY PRIMARY KEY NOT NULL,
address NVARCHAR(200) NOT NULL,
cost float NOT NULL,
square float NOT NULL,
number INT UNIQUE NOT NULL
)
2)
CREATE TABLE nedvishimost
(
id INT IDENTITY PRIMARY KEY NOT NULL,
uchastok_id INT NOT NULL,
CONSTRAINT fk_ned_uch FOREIGN KEY (uchastok_id)
       REFERENCES uchastok(id) ON DELETE CASCADE ON UPDATE CASCADE,
floor int NOT NULL,
flat int NULL,
square float NOT NULL,
cost float NOT NULL,
type NVARCHAR(100) NOT NULL,
number INT UNIQUE NOT NULL
)
CREATE TABLE dogovor
(
id INT IDENTITY PRIMARY KEY NOT NULL,
uchastok_id INT NOT NULL,
CONSTRAINT fk_dog_uch FOREIGN KEY (uchastok_id)
       REFERENCES uchastok(id),
nedvishimost_id INT NOT NULL,
CONSTRAINT fk_dog_ned FOREIGN KEY (nedvishimost_id)
       REFERENCES nedvishimost(id) ON DELETE CASCADE ON UPDATE CASCADE,
firma_id INT NOT NULL,
CONSTRAINT fk_dog_fir FOREIGN KEY (firma_id)
       REFERENCES firma(id) ON DELETE CASCADE ON UPDATE CASCADE,
type NVARCHAR(50) NOT NULL,
cost FLOAT NOT NULL,
pribil FLOAT NULL
)

CREATE TABLE HISTORY
(
    id INT IDENTITY PRIMARY KEY,
       dogovor_id INT NOT NULL,
       dogovor_pribil FLOAT NOT NULL,
```

```
)
CREATE TABLE firma
(
id INT IDENTITY PRIMARY KEY NOT NULL,
name NVARCHAR(100) NOT NULL,
director NVARCHAR(100) NOT NULL,
telephone NVARCHAR(12) NOT NULL,
inn NVARCHAR(20) NOT NULL
)
```

Запросы

Составной многотабличный запрос с CASE-выражением

Данные запросы включают в себя поля из основных таблиц, таким образом выдавая наиболее важную информацию, также с помощью CASE-выражения оценивается наценка, и определяется тип фирмы. Первый запрос служит для сделок с недвижимостью, второй – для сделок с земельными участками.

1 запрос

```
CREATE FUNCTION [dbo].[case_func2]()
RETURNS @my_temp1 TABLE
(
tip nvarchar(50),
zp float,
adda nvarchar(200),
kv INT,
fir nvarchar(100),
nacenka nvarchar(50),
tip_firmi nvarchar(20)
)
AS
BEGIN
WITH EMP_ctes(tip, zp, adda,kv,fir,nacenka,tip_firmi)
AS
(
select dogovor.type,dogovor.cost,uchastok.address,nedvishimost.flat,firma.name,
CASE
WHEN (dogovor.cost*120/nedvishimost.cost <0.5)AND(dogovor.type='Аренда') THEN
'Сильно  дешевле кадастра'
WHEN (dogovor.cost*120/nedvishimost.cost >= 0.5)AND(dogovor.cost*120/nedvishimost.cost <
1)AND(dogovor.type='Аренда') THEN 'Дешевле кадастра'
WHEN (dogovor.cost*120/nedvishimost.cost = 1)AND(dogovor.type='Аренда') THEN 'Ровно'
WHEN (dogovor.cost*120/nedvishimost.cost > 1)AND(dogovor.cost*120/nedvishimost.cost <=
2)AND(dogovor.type='Аренда') THEN 'Есть наценка'
WHEN (dogovor.cost*120/nedvishimost.cost > 2)AND(dogovor.cost*120/nedvishimost.cost <
5)AND(dogovor.type='Аренда') THEN 'Большая наценка'
WHEN (dogovor.cost*120/nedvishimost.cost >= 5)AND(dogovor.type='Аренда') THEN
'Очень  большая наценка'
WHEN (dogovor.cost/nedvishimost.cost <0.5)AND(dogovor.type='Покупка') THEN 'Сильно
дешевле кадастра'
WHEN (dogovor.cost/nedvishimost.cost >= 0.5)AND(dogovor.cost/nedvishimost.cost <
1)AND(dogovor.type='Покупка') THEN 'Дешевле кадастра'
WHEN (dogovor.cost/nedvishimost.cost = 1)AND(dogovor.type='Покупка') THEN 'Ровно'
WHEN (dogovor.cost/nedvishimost.cost > 1)AND(dogovor.cost/nedvishimost.cost <=
2)AND(dogovor.type='Покупка') THEN 'Есть наценка'
WHEN (dogovor.cost/nedvishimost.cost > 2)AND(dogovor.cost/nedvishimost.cost <
5)AND(dogovor.type='Покупка') THEN 'Большая наценка'
ELSE 'Очень большая
наценка'  END Nacenka,
CASE
WHEN (LEN(firma.inn) = 10) THEN
'Объединения'  ELSE 'ИП'
END type_of_firm
FROM dogovor JOIN nedvishimost ON nedvishimost.id=dogovor.nedvishimost_id JOIN uchastok
```

```
INSERT @my_temp1
SELECT tip, zp, adda,kv,fir,nacenka,tip_firmi FROM EMP_ctes
RETURN
END
```

2 запрос

```
CREATE FUNCTION [dbo].[testfunction3]()
RETURNS @my_temp TABLE
(
tip nvarchar(50),
zp float,
adda nvarchar(200),
fir nvarchar(100),
nacenka nvarchar(50),
tip_firmi nvarchar(20)
)
AS
BEGIN
WITH EMP_cte(tip, zp, adda,fir,nacenka,tip_firmi)
AS
(
select dogovor.type,dogovor.cost,uchastok.address,firma.name,
CASE
WHEN (dogovor.cost*120/uchastok.cost <0.5)AND(dogovor.type='Аренда') THEN 'Сильно
дешевле
кадастра'
WHEN (dogovor.cost*120/uchastok.cost >= 0.5)AND(dogovor.cost*120/uchastok.cost <
1)AND(dogovor.type='Аренда') THEN 'Дешевле кадастра'
WHEN (dogovor.cost*120/uchastok.cost = 1)AND(dogovor.type='Аренда') THEN 'Ровно'
WHEN (dogovor.cost*120/uchastok.cost > 1)AND(dogovor.cost*120/uchastok.cost <=
2)AND(dogovor.type='Аренда') THEN 'Есть наценка'
WHEN (dogovor.cost*120/uchastok.cost > 2)AND(dogovor.cost*120/uchastok.cost <
5)AND(dogovor.type='Аренда') THEN 'Большая наценка'
WHEN (dogovor.cost*120/uchastok.cost >= 5)AND(dogovor.type='Аренда') THEN 'Очень
большая  наценка'
WHEN (dogovor.cost/uchastok.cost <0.5)AND(dogovor.type='Покупка') THEN 'Сильно дешевле
кадастра'
WHEN (dogovor.cost/uchastok.cost >= 0.5)AND(dogovor.cost/uchastok.cost <
1)AND(dogovor.type='Покупка') THEN 'Дешевле кадастра'
WHEN (dogovor.cost/uchastok.cost = 1)AND(dogovor.type='Покупка') THEN 'Ровно'
WHEN (dogovor.cost/uchastok.cost > 1)AND(dogovor.cost/uchastok.cost <=
2)AND(dogovor.type='Покупка') THEN 'Есть наценка'
WHEN (dogovor.cost/uchastok.cost > 2)AND(dogovor.cost/uchastok.cost <
5)AND(dogovor.type='Покупка') THEN 'Большая наценка'
ELSE 'Очень большая наценка'
END Nacenka,
CASE
WHEN (LEN(firma.inn) = 10) THEN
'Объединения'  ELSE 'ИП'
END type_of_firm
FROM dogovor JOIN uchastok ON uchastok.id=dogovor.uchastok_id JOIN firma ON firma.id=
dogovor.firma_id WHERE dogovor.nedvishimost_id IS NULL)
INSERT @my_temp
SELECT tip, zp, adda,fir,nacenka,tip_firmi FROM EMP_cte
RETURN
END
```

Многотабличный VIEW, с возможностью его обновления

Данное представление имеет функцию обновления, содержит в себе полный адрес недвижимости, её характеристики и условия договора. Также эти данные имеет право обновлять “user”

```
CREATE OR ALTER VIEW ned_on_uch AS
	SELECT n.id AS Nid, u.address AS "Адрес", n.flat AS "Номер помещения",n.square
	AS
"Площадь",d.cost AS "Цена",d.type AS "Тип договора" FROM nedvishimost n JOIN uchastok
u  ON n.uchastok_id=u.id JOIN dogovor d ON d.nedvishimost_id=n.id

CREATE PROCEDURE update_view (@id INT,@address NVARCHAR(200),@flat INT,@square
float,@cost float,@type nvarchar(50))
AS
BEGIN
	UPDATE ned_on_uch SET "Адрес" = @address WHERE Nid = @ID
	UPDATE ned_on_uch SET "Номер помещения" = @flat,"Площадь" = @square WHERE Nid =
@ID
	UPDATE ned_on_uch SET "Цена" = @cost,"Тип договора"=@type WHERE Nid = @ID
END
GO
```

Запросы, содержащий подзапрос в разделах SELECT, FROM и WHERE

Первый запрос

Примеры запросов созданного на стороне клиента, подсчитывающий минимальное, среднее и максимальное значение кадастровой стоимости по всей таблице недвижимости, также выдаёт информацию по недвижимости дороже выбранной.

```
SELECT (SELECT address FROM uchastok u WHERE u.id=n.uchastok_id)AS
address,floor,flat,square,cost,pkdk.ned_min,ROUND(pkdk.ned_avg,2)AS ned_avg,pkdk.ned_max
FROM nedvishimost n,(SELECT min(x.cost)AS ned_min,ROUND(avg(x.cost),2)AS
ned_avg,max(x.cost)AS ned_max FROM nedvishimost x)as pkdk     WHERE cost > ALL(SELECT
cost  FROM nedvishimost WHERE id=14)
```

Второй запрос

Данный запрос продублирован, так ка подходит и к п3.3, так и к п.3.4. Содержит в себе информацию по деятельности фирм.

```
 SELECT f.name,(SELECT sum(d.cost) FROM dogovor d WHERE d.firma_id=f.id AND
d.type='Аренда')AS"Сумма аренды",(SELECT count(d.cost) FROM dogovor d WHERE
d.firma_id=f.id AND d.type='Аренда')AS"Кол-во аренды",(SELECT sum(d.cost) FROM dogovor d
WHERE d.firma_id=f.id AND d.type='Покупка')AS"Сумма покупок",(SELECT count(d.cost) FROM
dogovor d WHERE d.firma_id=f.id AND d.type='Покупка')AS "Кол-во покупок" ,doga_avg AS
"Общее кол-во" FROM firma f,(SELECT dd.firma_id AS fid, COUNT(*) AS doga_avg FROM
dogovor  dd GROUP BY dd.firma_id )as avg_dog WHERE avg_dog.fid=f.id AND(0 < ALL(SELECT
COUNT(*)  FROM dogovor d WHERE d.firma_id=f.id AND d.type='Аренда'))AND(1 < ALL(SELECT
COUNT(*)  FROM dogovor d WHERE d.firma_id=f.id AND d.type='Покупка'))
```

Коррелированные подзапросы

Данные подзапросы ссылаются на значения внешнего столбца, в первых двух случаях одной и той же таблицы находя самый дешёвый договор аренды для каждой фирмы и недвижимость с самой большой площадью для каждого участка. В третьем запросе взаимодействуют фирмы и договора. Данный запрос создан для статистики деятельности фирм (продублирован в предыдущей теме).

Первый запрос

```
SELECT *
  FROM dogovor pp
 WHERE pp.cost = (SELECT min(ppm.cost)
                    FROM dogovor ppm
                   WHERE ppm.firma_id = pp.firma_id AND ppm.type='Аренда')AND
pp.type='Аренда'
```

Второй запрос

```
SELECT *
  FROM nedvishimost pp
 WHERE pp.square = (SELECT max(ppm.square)
                     FROM nedvishimost ppm
                    WHERE ppm.uchastok_id = pp.uchastok_id)
```

Третий запрос

```
 SELECT f.name,(SELECT sum(d.cost) FROM dogovor d WHERE d.firma_id=f.id AND
d.type='Аренда')AS"Сумма аренды",(SELECT count(d.cost) FROM dogovor d WHERE
d.firma_id=f.id AND d.type='Аренда')AS"Кол-во аренды",(SELECT sum(d.cost) FROM dogovor d
WHERE d.firma_id=f.id AND d.type='Покупка')AS"Сумма покупок",(SELECT count(d.cost) FROM
dogovor d WHERE d.firma_id=f.id AND d.type='Покупка')AS "Кол-во покупок" ,doga_avg AS
"Общее кол-во" FROM firma f,(SELECT dd.firma_id AS fid, COUNT(*) AS doga_avg FROM
dogovor  dd GROUP BY dd.firma_id )as avg_dog WHERE avg_dog.fid=f.id AND(0 < ALL(SELECT
COUNT(*)  FROM dogovor d WHERE d.firma_id=f.id AND d.type='Аренда'))AND(1 < ALL(SELECT
COUNT(*)  FROM dogovor d WHERE d.firma_id=f.id AND d.type='Покупка'))
```

Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING

Пример запроса, обёрнутого в функцию, также используется дополнительная таблица и при выводе становятся не видны настоящие значения столбцов.

Создан для выборки по количеству недвижимости на участке.

```
CREATE function p35(@count INT)
RETURNS @my_temp2 TABLE
(
mesto nvarchar(200),
col_vo int
)
AS
BEGIN
WITH EMP_tst(mesto,col_vo)
AS
(
SELECT address,(SELECT     COUNT(*) As rep FROM nedvishimost n WHERE n.uchastok_id=u.id
GROUP BY uchastok_id HAVING(COUNT(*)>@count)) FROM uchastok u WHERE id in(SELECT
uchastok_id   FROM nedvishimost n  GROUP BY uchastok_id HAVING(COUNT(*)>@count))
)
INSERT @my_temp2
SELECT mesto,col_vo FROM EMP_tst
RETURN
END
```

Запросы, содержащий предикат ANY

Данный запрос также обёрнут в функцию. Его цель выбрать фирмы покупающие не дешевле определённой суммы.

```
ALTER function p36(@count FLOAT)
RETURNS @my_temp2 TABLE
(
nazvanie nvarchar(100),
glavniy nvarchar(100),
phone nvarchar(12),
buh nvarchar(20)
)
AS
BEGIN
WITH EMP_tstt(nazvanie,glavniy,phone,buh)
AS
(
SELECT name,director,telephone,inn FROM firma f WHERE id= any(SELECT firma_id FROM
dogovor WHERE cost >= @count AND type='Покупка')
)
INSERT @my_temp2
SELECT nazvanie,glavniy,phone,buh FROM EMP_tstt
RETURN
END
GO
```

Индексы

Созданы: уникальный индекс, индекс из NVARCHAR, составной индекс.

```
CREATE INDEX number ON uchastok(number)

CREATE INDEX firm ON firma(inn,name)


CREATE INDEX dogovors ON dogovor(nedvishimost_id,firma_id)
```

Триггеры

Создан триггер, реагирующий на обновление, удаление и изменения в таблице содержащей договора. Данный триггер переносит некоторые значения в таблицу истории операций с договорами. Также указывается операция.

```
CREATE TRIGGER doghis_DELETE
ON dogovor
AFTER DELETE
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Удалена'
FROM DELETED
GO
ALTER TRIGGER doghis_INSERT
ON dogovor
AFTER INSERT
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Добавлена'
FROM INSERTED
GO
CREATE TRIGGER doghis_UPDATE
ON dogovor
AFTER UPDATE
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Изменена'
FROM INSERTED
GO
```

Операции добавления, удаления и обновления реализовать в виде хранимых процедур или функций с параметрами для всех таблиц

```
CREATE PROCEDURE insert_uchastok(@address nvarchar(200), @square FLOAT,     @cost
FLOAT,  @number INT)
AS
BEGIN
INSERT INTO uchastok (address,square,cost,number) VALUES (@address,@square,@cost,@number)
END

CREATE PROCEDURE update_uchastok(@id INT, @address nvarchar(200), @square FLOAT,@cost
FLOAT,@number INT)
AS
BEGIN
UPDATE uchastok
SET address = @address,square=@square,number=@number,cost=@cost WHERE id = @id
END

CREATE PROCEDURE delete_uchastok(@id INT)
AS
BEGIN
DELETE FROM dogovor WHERE uchastok_id = @id
DELETE FROM uchastok WHERE id = @id
END

CREATE PROCEDURE insert_nedvishimost(@uchastok_id INT,@floor INT,@flat INT,@square
FLOAT,@cost FLOAT,@type NVARCHAR(100) , @number INT)
AS
BEGIN
INSERT INTO nedvishimost(uchastok_id,floor,flat,square,cost,type, number) VALUES
(@uchastok_id,@floor,@flat, @square,@cost, @type, @number)
END

CREATE PROCEDURE update_nedvishimost( @id INT,@uchastok_id INT,@floor INT,@flat
INT,@square FLOAT,@cost FLOAT,@type NVARCHAR(100) , @number INT)
AS
BEGIN
UPDATE nedvishimost
SET uchastok_id =
@uchastok_id,floor=@floor,flat=@flat,square=@square,cost=@cost,type=@type,number=@number
WHERE id = @id
END

CREATE PROCEDURE delete_nedvishimost ( @id INT)
AS
BEGIN
DELETE FROM nedvishimost WHERE id=@id
END

CREATE PROCEDURE insert_dogovor(@uchastok_id INT,@nedvishimost_id INT,@firma_id INT,@type
NVARCHAR(50),@cost FLOAT)
AS
BEGIN
INSERT INTO dogovor (uchastok_id,nedvishimost_id,firma_id,type,cost) VALUES
(@uchastok_id,@nedvishimost_id,@firma_id,@type,@cost)
END

CREATE PROCEDURE update_dogovor(@id INT,@uchastok_id INT,@nedvishimost_id INT,@firma_id
```

```
	UPDATE dogovor SET
uchastok_id=@uchastok_id,nedvishimost_id=@nedvishimost_id,firma_id=@firma_id,type=@type,c
ost=@cost WHERE id=@id
END

CREATE PROCEDURE delete_dogovor(@id INT)
AS
BEGIN
DELETE FROM dogovor WHERE id=@id
END

CREATE PROCEDURE insert_firma(@name NVARCHAR(100),@director NVARCHAR(100),@telephone
NVARCHAR(12),@inn NVARCHAR(20))
AS
BEGIN
	INSERT INTO firma (name,director,telephone,inn) VALUES
(@name,@director,@telephone,@inn)
END

CREATE PROCEDURE update_firma(@id INT,@name NVARCHAR(100),@director
NVARCHAR(100),@telephone NVARCHAR(12),@inn NVARCHAR(20))
AS
BEGIN
	UPDATE firma SET name=@name,director=@director,telephone=@telephone,inn=@inn WHERE
id=@id
END

CREATE PROCEDURE delete_firma(@id INT)
AS
BEGIN
DELETE FROM firma WHERE id=@id
END
```

Хранимая процедура или функция, состоящая из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана

Данная процедура помогает оценивает риск разрыва отношений с той или иной фирмой по их платежеспособности. Если фирма попадает под заданные (уже прописанные) критерии, то удаление через эту процедуру отменяется. Данная процедура не может отменить удаления напрямую через пользовательский интерфейс программы.

```
ALTER PROCEDURE try_del(@id INT)
AS
BEGIN
DECLARE @count INT
DECLARE @size FLOAT
DECLARE @cost FLOAT
SET @count=0
SET @size=0
SET @cost=0
BEGIN TRANSACTION
SET @count = (SELECT COUNT(*) FROM dogovor WHERE firma_id=@id GROUP BY firma_id)
SET @size = (SELECT SUM(pribil) FROM dogovor WHERE firma_id=@id AND type='Аренда' GROUP
BY firma_id)
SET @cost = (SELECT SUM(pribil) FROM dogovor WHERE firma_id=@id AND type='Покупка' GROUP
BY uchastok_id)
DELETE FROM firma WHERE id=@id
if @cost is null
SET @cost=0
if @size is null
SET @size=0
IF @count>2
ROLLBACK
ELSE IF @cost+@size*180>30000000
ROLLBACK
ELSE
COMMIT TRANSACTION
END
```

Курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы)

Данный курсор позволяет при желании пользователя пересчитать прибыль по прогрессивной шкале налогообложения (проценты придуманы). Данный курсор проходит по строкам таблице, проверяя полученный по договору доход, и меняет значение прибыли.

```
CREATE PROCEDURE trig
AS
BEGIN
DECLARE @curid integer
DECLARE @curpribil FLOAT
DECLARE @curcost FLOAT
DECLARE @curs CURSOR
SET @curs = CURSOR SCROLL FOR
SELECT id,cost FROM dogovor
OPEN @curs
FETCH NEXT FROM @curs INTO @curid,@curcost
WHILE @@FETCH_STATUS=0
BEGIN
if @curcost>100000000
UPDATE dogovor SET pribil=cost*0.8
else if @curcost>20000000
UPDATE dogovor SET pribil=cost*0.83
else if @curcost>5000000
UPDATE dogovor SET pribil=cost*0.85 WHERE id=@curid
FETCH NEXT FROM @curs INTO @curid,@curcost
END
CLOSE @curs
DEALLOCATE @curs
END
```

Скалярная и векторная функции

Скалярная функция

Возвращает адрес участка по его кадастровому номеру.

```
CREATE FUNCTION get_address(@number INT)
RETURNS NVARCHAR(200)
BEGIN
DECLARE @address NVARCHAR(200)
SELECT @address = address FROM uchastok
RETURN @address
END
```

Векторная функция

Возвращает таблицу выдающую самые прибыльные сделки по аренде и продаже. Так как запрос делается в таблицу операций, то это наибольший результат за всё время.

```
ALTER FUNCTION p92()
RETURNS @my_temp3 TABLE
(
mesto NVARCHAR(50),
col_vo int
)
AS
BEGIN
WITH EMP_tst3(mesto,col_vo)
AS
(
SELECT dogovor_type,max(dogovor_pribil) FROM HISTORY GROUP BY dogovor_type
)
INSERT @my_temp3
SELECT mesto,col_vo FROM EMP_tst3
RETURN
END
```

Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий

```
CREATE LOGIN Dbadmin
WITH PASSWORD = '2803';
CREATE USER Dbadmin
FOR LOGIN Dbadmin;
CREATE ROLE Dbadminrole;
ALTER ROLE Dbadminrole
ADD MEMBER Dbadmin;
CREATE LOGIN Dbuser
WITH PASSWORD = '1234';
CREATE USER Dbuser
FOR LOGIN Dbuser;
CREATE ROLE Dbuserrole
ALTER ROLE Dbuserrole
ADD MEMBER Dbuser
GRANT SELECT ON firma TO Dbuserrole
GRANT SELECT ON ned_on_uch TO Dbuserrole
GRANT SELECT ON nedvishimost TO Dbuserrole
GRANT SELECT ON dogovor TO Dbuserrole
GRANT SELECT ON uchastok TO Dbuserrole
GRANT EXECUTE ON testfunction3 TO Dbuserrole
GRANT EXECUTE ON case_func2 TO Dbuserrole
GRANT EXECUTE ON update_view TO Dbuserrole
GRANT EXECUTE ON p35 TO Dbuserrole
GRANT EXECUTE ON p36 TO Dbuserrole
GRANT EXECUTE ON trig TO Dbuserrole
GRANT EXECUTE ON get_address TO Dbuserrole
GRANT EXECUTE ON p92 TO Dbuserrole
GRANT CONTROL ON DATABASE::reestr TO Dbadmin
```

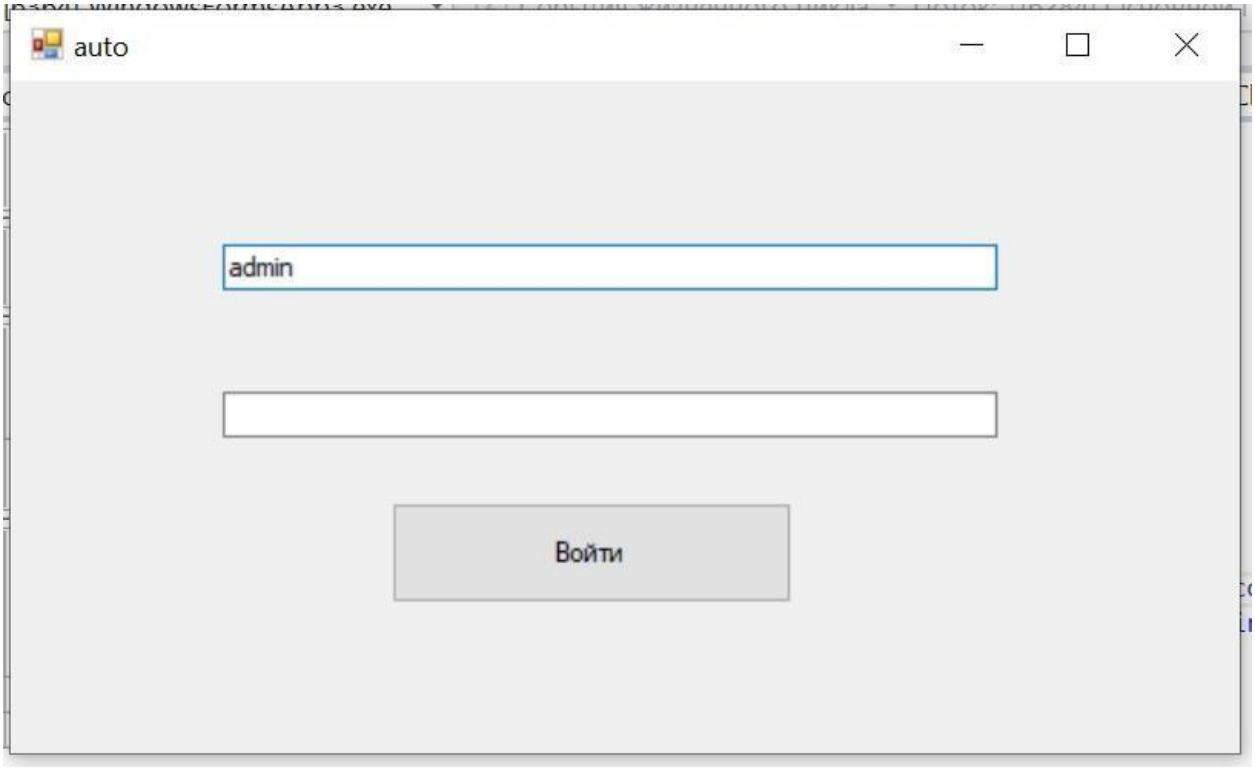
Список литературы

1. SQL Учебник[https://learndb.ru/articles]
2. Интерактивный учебник по SQL[http://www.sql-tutorial.ru/ru/book_correlated_subqueries.html]
3. Документация по Microsoft SQL[https://docs.microsoft.com/ru-ru/sql/?view=sql-server-ver16]
4. Windows Forms. Программирование на C# [http://csharpcoding.org/category/windows-forms/].
5. Язык программирования C#[https://metanit.com/sharp/]

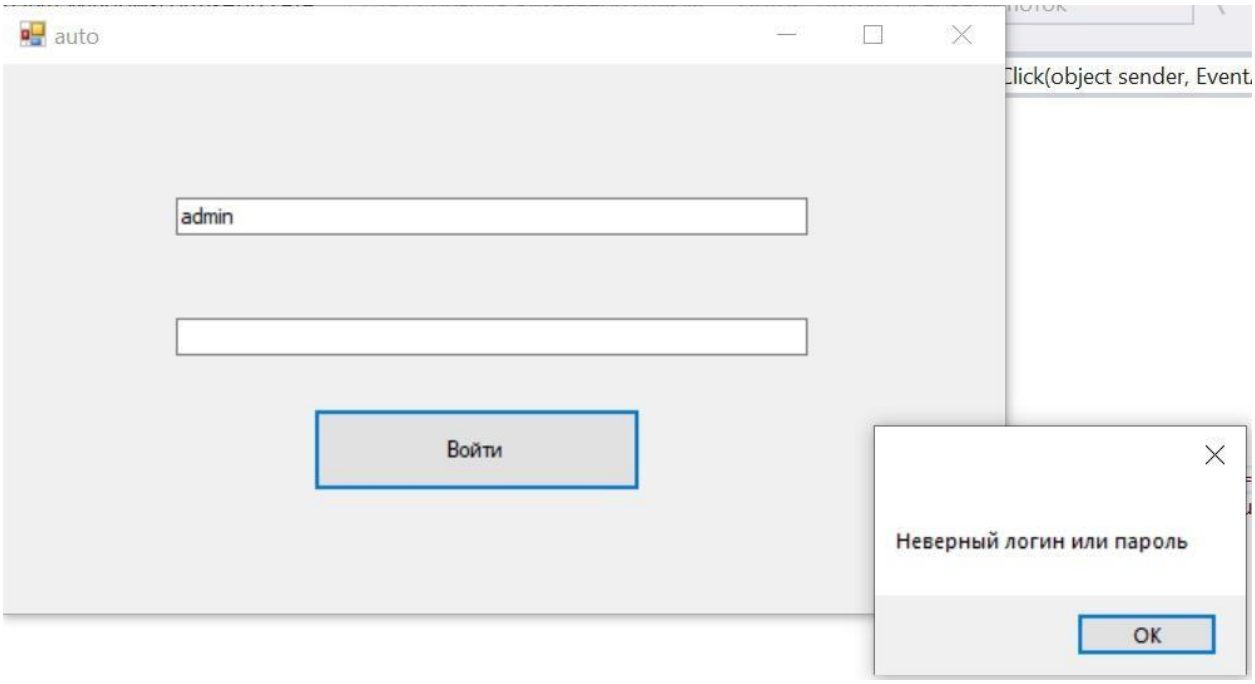
Приложение

Приложение А

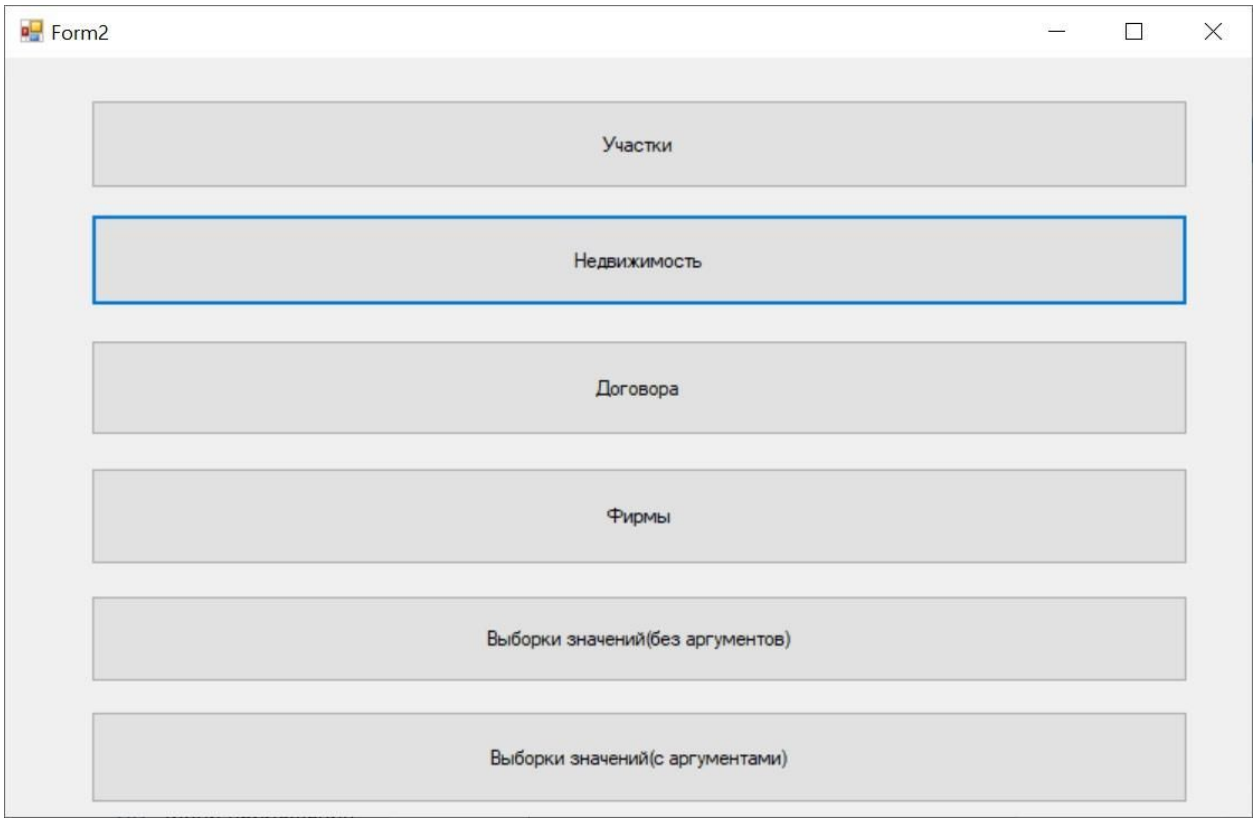
Вход в приложение



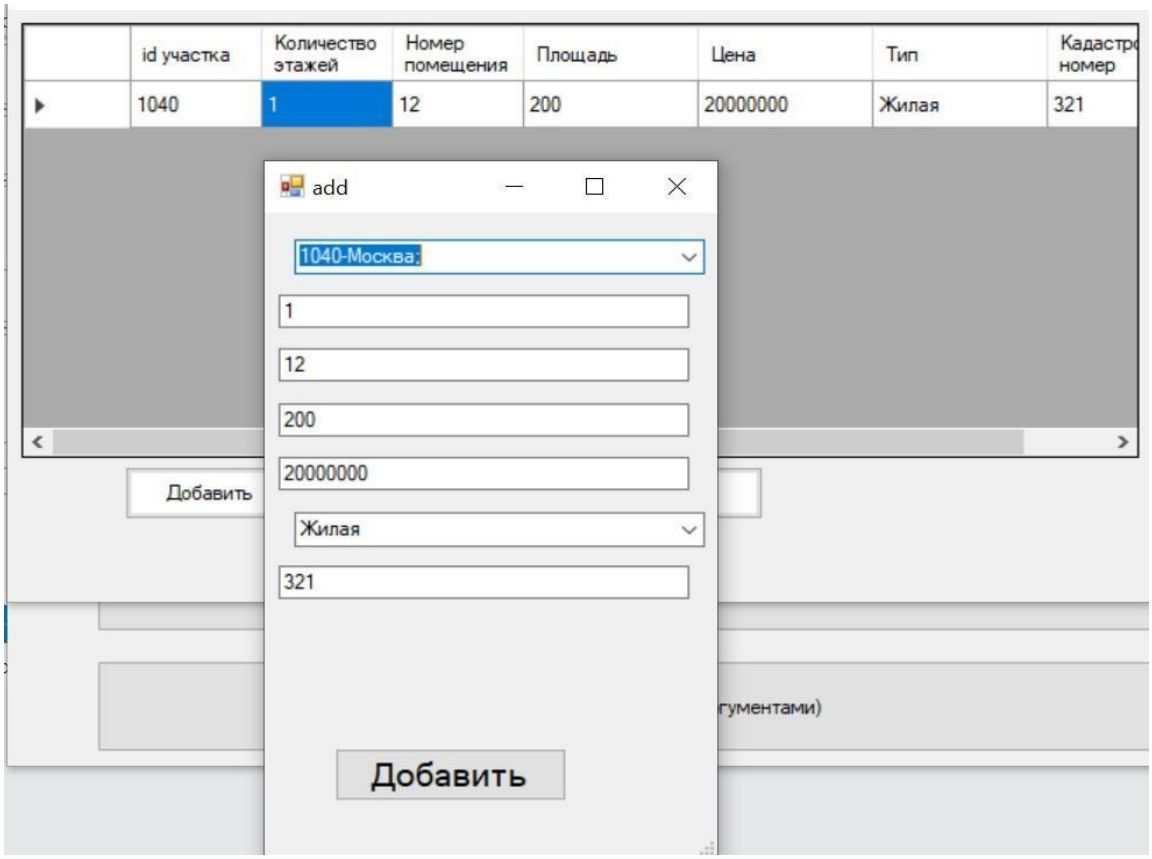
Контроль доступа в приложение



Главное меню



Изменение таблицы



Приложение Б

Листинг SQL

```
CREATE TABLE uchastok
(
id INT IDENTITY PRIMARY KEY NOT NULL,
address NVARCHAR(200) NOT NULL,
cost float NOT NULL,
square float NOT NULL,
number INT UNIQUE NOT NULL
)
GO
CREATE TABLE nedvishimost
(
id INT IDENTITY PRIMARY KEY NOT NULL,
uchastok_id INT NOT NULL,
CONSTRAINT fk_ned_uch FOREIGN KEY (uchastok_id)
      REFERENCES uchastok(id) ON DELETE CASCADE ON UPDATE CASCADE,
floor int NOT NULL,
flat int NULL,
square float NOT NULL,
cost float NOT NULL,
type NVARCHAR(100) NOT NULL,
number INT UNIQUE NOT NULL
)
GO
CREATE TABLE dogovor
(
id INT IDENTITY PRIMARY KEY NOT NULL,
uchastok_id INT NOT NULL,
CONSTRAINT fk_dog_uch FOREIGN KEY (uchastok_id)
      REFERENCES uchastok(id),
nedvishimost_id INT NOT NULL,
CONSTRAINT fk_dog_ned FOREIGN KEY (nedvishimost_id)
      REFERENCES nedvishimost(id) ON DELETE CASCADE ON UPDATE CASCADE,
firma_id INT NOT NULL,
CONSTRAINT fk_dog_fir FOREIGN KEY (firma_id)
      REFERENCES firma(id) ON DELETE CASCADE ON UPDATE CASCADE,
type NVARCHAR(50) NOT NULL,
cost FLOAT NOT NULL,
pribil FLOAT NULL
)
GO
CREATE TABLE firma
(
id INT IDENTITY PRIMARY KEY NOT NULL,
name NVARCHAR(100) NOT NULL,
director NVARCHAR(100) NOT NULL,
telephone NVARCHAR(12) NOT NULL,
inn NVARCHAR(20) NOT NULL
)
GO
CREATE PROCEDURE insert_uchastok(@address nvarchar(200), @square FLOAT,
@cost FLOAT, @number INT)
AS
BEGIN
```

```
CREATE PROCEDURE update_uchastok(@id INT, @address nvarchar(200), @square
FLOAT,@cost FLOAT,@number INT)
AS
BEGIN
UPDATE uchastok
SET address = @address,square=@square,number=@number,cost=@cost WHERE id =
@id
END
GO
CREATE PROCEDURE delete_uchastok(@id INT)
AS
BEGIN
DELETE FROM dogovor WHERE uchastok_id = @id
DELETE FROM uchastok WHERE id = @id
END
GO
CREATE PROCEDURE insert_nedvishimost(@uchastok_id INT,@floor INT,@flat
INT,@square FLOAT,@cost FLOAT,@type NVARCHAR(100) , @number INT)
AS
BEGIN
INSERT INTO nedvishimost(uchastok_id,floor,flat,square,cost,type, number)
VALUES (@uchastok_id,@floor,@flat, @square,@cost, @type, @number)
END
GO
CREATE PROCEDURE update_nedvishimost( @id INT,@uchastok_id INT,@floor
INT,@flat INT,@square FLOAT,@cost FLOAT,@type NVARCHAR(100) , @number INT)
AS
BEGIN
UPDATE nedvishimost
SET uchastok_id =
@uchastok_id,floor=@floor,flat=@flat,square=@square,cost=@cost,type=@type,num
ber=@number WHERE id = @id
END
GO
CREATE PROCEDURE delete_nedvishimost ( @id INT)
AS
BEGIN
DELETE FROM nedvishimost WHERE id=@id
END
GO
CREATE PROCEDURE insert_dogovor(@uchastok_id INT,@nedvishimost_id
INT,@firma_id INT,@type NVARCHAR(50),@cost FLOAT)
AS
BEGIN
INSERT INTO dogovor (uchastok_id,nedvishimost_id,firma_id,type,cost) VALUES
(@uchastok_id,@nedvishimost_id,@firma_id,@type,@cost)
END
GO
CREATE PROCEDURE update_dogovor(@id INT,@uchastok_id INT,@nedvishimost_id
INT,@firma_id INT,@type NVARCHAR(50),@cost FLOAT)
AS
BEGIN
      UPDATE dogovor SET
uchastok_id=@uchastok_id,nedvishimost_id=@nedvishimost_id,firma_id=@firma_id,
type=@type,cost=@cost WHERE id=@id
END
GO
CREATE PROCEDURE delete_dogovor(@id INT)
AS
```

```
CREATE PROCEDURE insert_firma(@name NVARCHAR(100),@director
NVARCHAR(100),@telephone  NVARCHAR(12),@inn  NVARCHAR(20))
AS
BEGIN
      INSERT INTO firma (name,director,telephone,inn) VALUES
(@name,@director,@telephone,@inn)
END
GO
CREATE PROCEDURE update_firma(@id INT,@name NVARCHAR(100),@director
NVARCHAR(100),@telephone NVARCHAR(12),@inn NVARCHAR(20))
AS
BEGIN
      UPDATE firma SET
name=@name,director=@director,telephone=@telephone,inn=@inn WHERE id=@id
END
GO
CREATE PROCEDURE delete_firma(@id INT)
AS
BEGIN
DELETE FROM firma WHERE id=@id
END
GO
CREATE FUNCTION [dbo].[case_func2]()
RETURNS @my_temp1 TABLE
(
tip nvarchar(50),
zp float,
adda nvarchar(200),
kv INT,
fir nvarchar(100),
nacenka nvarchar(50),
tip_firmi nvarchar(20)
)
AS
BEGIN
WITH EMP_ctes(tip, zp, adda,kv,fir,nacenka,tip_firmi)
AS
(
select
dogovor.type,dogovor.cost,uchastok.address,nedvishimost.flat,firma.name,
CASE
WHEN (dogovor.cost*120/nedvishimost.cost <0.5)AND(dogovor.type='Аренда') THEN
'Сильно дешевле кадастра'
WHEN (dogovor.cost*120/nedvishimost.cost >=
0.5)AND(dogovor.cost*120/nedvishimost.cost < 1)AND(dogovor.type='Аренда')
THEN 'Дешевле кадастра'
WHEN (dogovor.cost*120/nedvishimost.cost = 1)AND(dogovor.type='Аренда') THEN
'Ровно'
WHEN (dogovor.cost*120/nedvishimost.cost >
1)AND(dogovor.cost*120/nedvishimost.cost <= 2)AND(dogovor.type='Аренда') THEN
'Есть наценка'
WHEN (dogovor.cost*120/nedvishimost.cost >
2)AND(dogovor.cost*120/nedvishimost.cost < 5)AND(dogovor.type='Аренда') THEN
'Большая наценка'
WHEN (dogovor.cost*120/nedvishimost.cost >= 5)AND(dogovor.type='Аренда') THEN
'Очень большая наценка'
WHEN (dogovor.cost/nedvishimost.cost <0.5)AND(dogovor.type='Покупка') THEN
'Сильно дешевле кадастра'
WHEN (dogovor.cost/nedvishimost.cost >=
```

```
WHEN (dogovor.cost/nedvishimost.cost > 1)AND(dogovor.cost/nedvishimost.cost
<= 2)AND(dogovor.type='Покупка') THEN 'Есть наценка'
WHEN (dogovor.cost/nedvishimost.cost > 2)AND(dogovor.cost/nedvishimost.cost <
5)AND(dogovor.type='Покупка') THEN 'Большая наценка'
ELSE 'Очень большая наценка'
END Nacenka,
CASE
WHEN (LEN(firma.inn) = 10) THEN 'Объединения'
ELSE 'ИП'
END type_of_firm
FROM dogovor JOIN nedvishimost ON nedvishimost.id=dogovor.nedvishimost_id  JOIN
uchastok ON uchastok.id=nedvishimost.uchastok_id JOIN firma ON firma.id=
dogovor.firma_id   WHERE dogovor.uchastok_id IS NULL
)
INSERT @my_temp1
SELECT tip, zp, adda,kv,fir,nacenka,tip_firmi FROM EMP_ctes
RETURN
END
GO
CREATE FUNCTION [dbo].[testfunction3]()
RETURNS @my_temp TABLE
(
tip nvarchar(50),
zp float,
adda nvarchar(200),
fir nvarchar(100),
nacenka nvarchar(50),
tip_firmi nvarchar(20)
)
AS
BEGIN
WITH EMP_cte(tip, zp, adda,fir,nacenka,tip_firmi)
AS
(
select dogovor.type,dogovor.cost,uchastok.address,firma.name,
CASE
WHEN (dogovor.cost*120/uchastok.cost <0.5)AND(dogovor.type='Аренда') THEN
'Сильно дешевле кадастра'
WHEN (dogovor.cost*120/uchastok.cost >=
0.5)AND(dogovor.cost*120/uchastok.cost < 1)AND(dogovor.type='Аренда') THEN
'Дешевле кадастра'
WHEN (dogovor.cost*120/uchastok.cost = 1)AND(dogovor.type='Аренда') THEN
'Ровно'
WHEN (dogovor.cost*120/uchastok.cost > 1)AND(dogovor.cost*120/uchastok.cost
<= 2)AND(dogovor.type='Аренда') THEN 'Есть наценка'
WHEN (dogovor.cost*120/uchastok.cost > 2)AND(dogovor.cost*120/uchastok.cost <
5)AND(dogovor.type='Аренда') THEN 'Большая наценка'
WHEN (dogovor.cost*120/uchastok.cost >= 5)AND(dogovor.type='Аренда') THEN
'Очень большая наценка'
WHEN (dogovor.cost/uchastok.cost <0.5)AND(dogovor.type='Покупка') THEN
'Сильно дешевле кадастра'
WHEN (dogovor.cost/uchastok.cost >= 0.5)AND(dogovor.cost/uchastok.cost <
1)AND(dogovor.type='Покупка') THEN 'Дешевле кадастра'
WHEN (dogovor.cost/uchastok.cost = 1)AND(dogovor.type='Покупка') THEN 'Ровно'
WHEN (dogovor.cost/uchastok.cost > 1)AND(dogovor.cost/uchastok.cost <=
2)AND(dogovor.type='Покупка') THEN 'Есть наценка'
WHEN (dogovor.cost/uchastok.cost > 2)AND(dogovor.cost/uchastok.cost <
5)AND(dogovor.type='Покупка') THEN 'Большая наценка'
ELSE 'Очень большая наценка'
```

```
FROM dogovor JOIN uchastok ON uchastok.id=dogovor.uchastok_id JOIN firma ON
firma.id= dogovor.firma_id WHERE dogovor.nedvishimost_id IS NULL)
INSERT @my_temp
SELECT tip, zp, adda,fir,nacenka,tip_firmi FROM EMP_cte
RETURN
END
GO
CREATE OR ALTER VIEW ned_on_uch AS
      SELECT n.id AS Nid, u.address AS "Адрес", n.flat AS "Номер
помещения",n.square AS "Площадь",d.cost AS "Цена",d.type AS "Тип договора"
FROM nedvishimost n JOIN uchastok u ON n.uchastok_id=u.id JOIN dogovor d ON
d.nedvishimost_id=n.id
GO
CREATE PROCEDURE update_view (@id INT,@address NVARCHAR(200),@flat
INT,@square float,@cost float,@type nvarchar(50))
AS     UPDATE      ned_on_uch SET "Адрес" = @address WHERE Nid = @ID
BEGIN
WHERE Nid = @ID
      UPDATE      ned_on_uch SET "Номер помещения" = @flat,"Площадь" = @square
      UPDATE      ned_on_uch SET "Цена" = @cost,"Тип договора"=@type WHERE Nid
= @ID
END
GO
CREATE function p35(@count INT)
RETURNS @my_temp2 TABLE
(
mesto nvarchar(200),
col_vo int
)
AS
BEGIN
WITH EMP_tst(mesto,col_vo)
AS
(
SELECT address,(SELECT   COUNT(*) As rep FROM nedvishimost n WHERE
n.uchastok_id=u.id GROUP BY uchastok_id HAVING(COUNT(*)>@count)) FROM
uchastok u WHERE id in(SELECT  uchastok_id  FROM nedvishimost n      GROUP
BY
uchastok_id HAVING(COUNT(*)>@count))
)
INSERT @my_temp2
SELECT mesto,col_vo FROM EMP_tst
RETURN
END
GO
ALTER function p36(@count FLOAT)
RETURNS @my_temp2 TABLE
(
nazvanie nvarchar(100),
glavniy nvarchar(100),
phone nvarchar(12),
buh nvarchar(20)
)
AS
BEGIN
WITH EMP_tstt(nazvanie,glavniy,phone,buh)
AS
(
SELECT name,director,telephone,inn FROM firma f WHERE id= any(SELECT firma_id
FROM dogovor WHERE cost >= @count AND type='Покупка')
)
INSERT @my_temp2
```

```
END
GO
CREATE INDEX number ON uchastok(number)
GO
CREATE INDEX firm ON firma(inn,name)
GO
CREATE INDEX dogovors ON dogovor(nedvishimost_id,firma_id)
GO
CREATE TABLE HISTORY
(
    id INT IDENTITY PRIMARY KEY,
      dogovor_id INT NOT NULL,
      dogovor_pribil FLOAT NOT NULL,
      dogovor_type NVARCHAR(50) NOT NULL,
      operation NVARCHAR(50) NOT NULL,
      CreateAt DATETIME NOT NULL DEFAULT GETDATE()
)
GO
CREATE TRIGGER doghis_DELETE
ON dogovor
AFTER DELETE
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Удалена'
FROM DELETED
GO
ALTER TRIGGER doghis_INSERT
ON dogovor
AFTER INSERT
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Добавлена'
FROM INSERTED
GO
CREATE TRIGGER doghis_UPDATE
ON dogovor
AFTER UPDATE
AS
INSERT INTO history(dogovor_id,dogovor_pribil,dogovor_type,operation)
SELECT id,pribil,type,'Изменена'
FROM INSERTED
GO
ALTER PROCEDURE try_del(@id INT)
AS
BEGIN
DECLARE @count INT
DECLARE @size FLOAT
DECLARE @cost FLOAT
SET @count=0
SET @size=0
SET @cost=0
BEGIN TRANSACTION
SET @count = (SELECT COUNT(*) FROM dogovor WHERE firma_id=@id GROUP BY
firma_id)
SET @size = (SELECT SUM(pribil) FROM dogovor WHERE firma_id=@id AND
type='Аренда' GROUP BY firma_id)
SET @cost = (SELECT SUM(pribil) FROM dogovor WHERE firma_id=@id AND
type='Покупка' GROUP BY uchastok_id)
DELETE FROM firma WHERE id=@id
if @cost is null
```

```
ROLLBACK
ELSE IF @cost+@size*180>30000000
ROLLBACK
ELSE
COMMIT TRANSACTION
END
GO
CREATE PROCEDURE trig
AS
BEGIN
DECLARE @curid integer
DECLARE @curpribil FLOAT
DECLARE @curcost FLOAT
DECLARE @curs CURSOR
SET @curs = CURSOR SCROLL FOR
SELECT id,cost FROM dogovor
OPEN @curs
FETCH NEXT FROM @curs INTO @curid,@curcost
WHILE @@FETCH_STATUS=0
BEGIN
if @curcost>100000000
UPDATE dogovor SET pribil=cost*0.8
else if @curcost>20000000
UPDATE dogovor SET pribil=cost*0.83
else if @curcost>5000000
UPDATE dogovor SET pribil=cost*0.85 WHERE id=@curid
FETCH NEXT FROM @curs INTO @curid,@curcost
END
CLOSE @curs
DEALLOCATE @curs
END
GO
CREATE FUNCTION get_address(@number INT)
RETURNS NVARCHAR(200)
BEGIN
DECLARE @address NVARCHAR(200)
SELECT @address = address FROM uchastok
RETURN @address
END
GO
ALTER FUNCTION p92()
RETURNS @my_temp3 TABLE
(
mesto NVARCHAR(50),
col_vo int
)
AS
BEGIN
WITH EMP_tst3(mesto,col_vo)
AS
(
SELECT dogovor_type,max(dogovor_pribil) FROM HISTORY GROUP BY dogovor_type
)
INSERT @my_temp3
SELECT mesto,col_vo FROM EMP_tst3
RETURN
END
GO
CREATE LOGIN Dbadmin
```

```
ADD MEMBER Dbadmin;
CREATE LOGIN Dbuser
WITH PASSWORD = '1234';
CREATE USER Dbuser
FOR LOGIN Dbuser;
CREATE ROLE Dbuserrole
ALTER ROLE Dbuserrole
ADD MEMBER Dbuser
GRANT SELECT ON firma TO Dbuserrole
GRANT SELECT ON ned_on_uch TO Dbuserrole
GRANT SELECT ON nedvishimost TO Dbuserrole
GRANT SELECT ON dogovor TO Dbuserrole
GRANT SELECT ON uchastok TO Dbuserrole
GRANT EXECUTE ON testfunction3 TO Dbuserrole
GRANT EXECUTE ON case_func2 TO Dbuserrole
GRANT EXECUTE ON update_view TO Dbuserrole
GRANT EXECUTE ON p35 TO Dbuserrole
GRANT EXECUTE ON p36 TO Dbuserrole
GRANT EXECUTE ON trig TO Dbuserrole
GRANT EXECUTE ON get_address TO Dbuserrole
GRANT EXECUTE ON p92 TO Dbuserrole
GRANT CONTROL ON DATABASE::reestr TO Dbadmin
GO
```

Листинг приложения

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp3
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new auto());
        }
    }
}
```

Program.cs

```
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;
```

```
namespace WindowsFormsApp3
{
    public partial class auto : Form
    {
        public auto()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "admin") { if (textBox2.Text == "2803") { string con =
$@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
Form2 f = new Form2(con); f.Show(); this.Hide(); } else { MessageBox.Show("Неверный логин
или пароль"); } }
            else if (textBox1.Text == "user") { if (textBox2.Text == "1234") { string con
= $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated Security=False;User
Id=Dbuser;Password=1234;MultipleActiveResultSets=True;TrustServerCertificate=true;";
Form2 f = new Form2(con); f.Show(); this.Hide(); } else { MessageBox.Show("Неверный логин
или пароль"); } }
        }
    }
}
```

auto.cs

```
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 WindowsFormsApp3
{
    public partial class Form2 : Form
    {
        Form1 f;
        private string conect;
        public Form2(string con)
        {
            conect = con;
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            /*string sqls = "SELECT* FROM uchastok";
            string[] names = new string[] {"Адрес", "Площадь (кв.м)", "Кадастровая
цена",  "Кадастровый номер" };
            int[] shirina = new int[] { 275, 100, 100, 100 };
            string connectionstring = $@"Data Source=.\SQLEXPRESS;Initial
Catalog=reestr;Integrated Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            string[] myArray = null;
            string table = "uchastok";*/
            Standart_table table1 = new Standart_table();
            table1.names = new string[] { "Адрес", "Площадь (кв.м)", "Кадастровая
цена",  "Кадастровый номер" };
```

```
            table1.table = "uchastok";
            table1.foreigns = new List<foreign>();
            table1.combos = new List<combo>();
            table1.unique = 4;
            String[] tipd = new String[] { "string", "double", "double", "int" };
            Console.WriteLine(tipd[0]);
            Form1 f = new Form1(table1, tipd);
            f.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Standart_table table2 = new Standart_table();
            table2.names = new string[] { "id участка", "Количество этажей", "Номер
помещения", "Площадь", "Цена", "Тип", "Кадастровый номер" };
            table2.shirina = new int[] { 75, 75, 75, 100, 100, 100, 100 };
            table2.sqlcon = conect;
            table2.sqlq = "SELECT * FROM nedvishimost";
            table2.tip = "standart";
            table2.table = "nedvishimost";
            table2.foreigns = new List<foreign>();
            table2.foreigns.Add(new foreign(0, "select id,address from uchastok"));
            table2.combos = new List<combo>();
            List<string> str = new List<string> { "Жилая", "Полужилая", "Нежилая",
"Особые условия" };
            table2.combos.Add(new combo(5, str));
            table2.unique = 7;
            String[] tip = new String[] { "int", "int", "int", "double", "double",
"string", "int" };
            Form1 f = new Form1(table2, tip);
            f.Show();
        }

        private void Form2_Load(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            Standart_table table3 = new Standart_table();
            table3.names = new string[] { "id участка", "id недвижимости", "id
фирмы",  "Тип", "Цена", "Прибыль" };
            table3.shirina = new int[] { 75, 75, 75, 100, 100, 100, };
            table3.sqlcon = conect;
            table3.sqlq = "SELECT * FROM dogovor";
            table3.tip = "standart";
            table3.table = "dogovor";
            table3.foreigns = new List<foreign>();
            table3.foreigns.Add(new foreign(0, "select id,address from uchastok"));
            table3.foreigns.Add(new foreign(1, "select n.id,u.address,n.flat from
uchastok u join nedvishimost n ON u.id=n.uchastok_id"));
            table3.foreigns.Add(new foreign(2, "select id,name from firma"));
            table3.combos = new List<combo>();
            List<string> str = new List<string> { "Покупка", "Аренда" };
            table3.combos.Add(new combo(3, str));
            String[] tip = new String[] { "int", "int", "int", "string", "double",
"double" };
            Form1 f = new Form1(table3, tip);
            f.Show();
```

```
            table1.names = new string[] { "Название", "Директор", "Телефон", "ИНН"
            };  table1.shirina = new int[] { 150, 150, 100, 100 };
            table1.sqlcon = conect;
            table1.sqlq = "SELECT * FROM firma";
            table1.tip = "standart";
            table1.table = "firma";
            table1.foreigns = new List<foreign>();
            table1.combos = new List<combo>();
            table1.unique = 4;
            String[] tip = new String[] { "string", "string", "string", "string" };
            Form1 f = new Form1(table1, tip);
            f.Show();
        }

        private void button5_Click(object sender, EventArgs e)
        {

            Form4 f = new Form4(conect);
            f.Show();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            select_usl f = new select_usl(conect);
            f.Show();
        }
    }
}
```

form2.cs

```
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;
using Microsoft.Data.SqlClient;
using System.Reflection;
namespace WindowsFormsApp3
{
    public partial class Form1 : Form
    {
        private string[] val;
        private List<int> id;
        private string[] tip;
        private List<int> uniqs;
        public List<int> frg;
        public List<int> chastfrg;
        private Standart_table table;
        public Form1(Standart_table table1,string[] tipe)
        {
            table = table1;
            tip = tipe;
            int x = 0;
            if (table.names.Length > 7)
            {
                x = (table.names.Length - 7) * 50;
                int h = 600;
```

```
            dataGridView1.Size = new System.Drawing.Size(w1+x, h1 + x);
        }
        Button btn = new Button();
        btn.Location = new System.Drawing.Point(100, 420 + x);
        btn.Size = new System.Drawing.Size(150, 50);
        btn.Text = "Добавить";
        btn.Name = "btn";
        btn.BackColor = Color.White;
        btn.Click += new EventHandler(btn_Click);
        this.Controls.Add(btn);
        Button btn1 = new Button();
        btn1.Location = new System.Drawing.Point(300, 420 + x);
        btn1.Size = new System.Drawing.Size(150, 50);
        btn1.Text = "Изменить";
        btn1.Name = "btn";
        btn1.BackColor = Color.White;
        btn1.Click += new EventHandler(btn_Click1);
        this.Controls.Add(btn1);
        Button btn2 = new Button();
        btn2.Location = new System.Drawing.Point(500, 420 + x);
        btn2.Size = new System.Drawing.Size(150, 50);
        btn2.Text = "Удалить";
        btn2.Name = "btn";
        btn2.BackColor = Color.White;
        btn2.Click += new EventHandler(btn_Click2);
        this.Controls.Add(btn2);
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        dataGridView1.AllowUserToAddRows = false;
        dataGridView1.AllowUserToDeleteRows = false;
        dataGridView1.ReadOnly = true;
        dataGridView1.AllowUserToOrderColumns = false;
        DataGridViewTextBoxColumn[] dgvc;
        dgvc = new DataGridViewTextBoxColumn[table.names.Length];
        for (int i = 0; i < table.names.Length; i++)
        {
            dgvc[i] = new DataGridViewTextBoxColumn();
            dgvc[i].HeaderText = table.names[i];
            dgvc[i].Name = "col" + i;
            dgvc[i].Width = table.shirina[i];
            Console.WriteLine(table.shirina[i]);
            dgvc[i].SortMode = DataGridViewColumnSortMode.NotSortable;
            dataGridView1.Columns.Add(dgvc[i]);
        }
        using (SqlConnection connection = new SqlConnection(table.sqlcon))
        {
            connection.Open();
            uniqs = new List<int>();
            SqlCommand command = new SqlCommand();
            command.CommandText = table.sqlq;
            command.Connection = connection;
            SqlDataReader reader = command.ExecuteReader();
            int count = reader.FieldCount;
            string[] s; s = new string[reader.FieldCount - 1];
            id = new List<int>();
            int j = 1;
            while (reader.Read())
```

```
                }
                dataGridView1.Rows.Add(s);
                j++;
            }
            List<string> value = new List<string>();
            frg = new List<int>();
            string curval;
            for (int i = 0; i < table.foreigns.Count; i++)
            {
            curval = "";
            SqlCommand fcom = new SqlCommand(table.foreigns[i].fsqlq, connection);
                Console.WriteLine(table.foreigns[i].fsqlq);
            SqlDataReader fk = fcom.ExecuteReader();
            while (fk.Read())
            {
                curval =  fk[0].ToString()+"-";
                for(int k = 1; k < fk.FieldCount; k++)
                {
                   curval+=fk[k].ToString();
                   if (i != fk.FieldCount - 1) { curval += ";"; }
                }
            value.Add(curval);
            }
            table.combos.Add(new combo(i, value));
            frg.Add(table.foreigns[i].id);
            value = new List<string>();
            }
        }
        foreach(combo x in table.combos)
        {
            Console.WriteLine(x.id);
            foreach(string str in x.values)
            {
                Console.WriteLine(str);
            }
        }
    }
    private void btn_Click(object sender, EventArgs e)
    {
        string commandq;
        val = null;
        bool itk = false;
        if (table.table == "dogovor") { itk = true; }
        form3 fm2 = new form3(table.names, table.combos, tip,val,frg,itk);
        fm2.Size = new System.Drawing.Size(275, 50 * (table.names.Length + 1) + 25);
        fm2.ShowDialog();
        if (fm2.DialogResult == DialogResult.OK)
        {
            commandq = "EXECUTE insert_" + table.table + " " + fm2.result;
            if (itk) { commandq = commandq.Substring(0, commandq.Length - 1); }
            using (SqlConnection connection = new SqlConnection(table.sqlcon))
            {
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.CommandText = commandq;
                Console.WriteLine(commandq);
                command.Connection = connection;
                if (command.ExecuteNonQuery() > 0)
                {
                    Form1 f = new Form1(table.tip);
```

```
        }
        private void btn_Click1(object sender, EventArgs e)
        {
            string commandq="";
            val = new string[table.names.Length];
            if (dataGridView1.SelectedCells.Count > 0)
            {
                int idr = dataGridView1.SelectedCells[0].RowIndex;
                for (int i = 0; i < table.names.Length; i++)
                {
                    val[i] = dataGridView1.Rows[idr].Cells[i].Value.ToString();
                }
                bool itk = false;
                if (table.table == "dogovor") { itk = true; }
                form3 fm2 = new form3(table.names, table.combos, tip, val,frg,itk);
                fm2.Size = new System.Drawing.Size(275, 50 * (table.names.Length + 1) +
25);
                fm2.ShowDialog();
                if (fm2.DialogResult == DialogResult.OK)
                {
                    commandq = "EXECUTE update_" + table.table + " "+id[idr]+"," +
fm2.result;
                    if (itk) { commandq = commandq.Substring(0, commandq.Length - 1);}
                    Console.WriteLine(commandq);
                    using (SqlConnection connection = new SqlConnection(table.sqlcon))
                    {
                        connection.Open();
                        SqlCommand command = new SqlCommand();
                        command.CommandText = commandq;
                        command.Connection = connection;
                        if (command.ExecuteNonQuery() > 0)
                        {
                            Form1 f = new Form1(table,tip);
                            f.Show();
                            this.Close();
                        }
                    }
                }
            }
        }
        private void btn_Click2(object sender, EventArgs e)
        {
            if (dataGridView1.SelectedCells.Count > 0)
            {
                int idr = id[dataGridView1.SelectedCells[0].RowIndex];
                string comandq = "EXECUTE delete_"+table.table+" "+idr;
                using (SqlConnection connection = new SqlConnection(table.sqlcon))
                {
                    Console.WriteLine(comandq);
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.CommandText = comandq;
                    command.Connection = connection;
                    if (command.ExecuteNonQuery() > 0)
                    {
                        Form1 f = new Form1(table,tip);
                        f.Show();
                        this.Close();
                    }
                }
```

form1.cs

```
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 WindowsFormsApp3
{
    public partial class form3 : Form
    {
        private ComboBox[] cb;
        private TextBox[] tb;
        public string result;
        private List<combo> combos;
        private List<int> forg;
        private string[] names;
        private string[] tip;
        private int c;
        private int t;
        private bool iss;
        private bool itke;
        private List<int> uniqe;
        public form3(string[]nam,List<combo> com,string[] tipes,string[]valu,List<int>
frg,bool itk)
        {
            this.DialogResult = DialogResult.No;
            this.Size= new System.Drawing.Size(250, 50*(nam.Length+2));
            forg = frg;
            itke=itk;
            combos = com;
            names = nam;
            tip = tipes;
            Console.WriteLine(tip[0]);
             c=0;
             t=0;
             iss = false;
            cb = new ComboBox[com.Count];
            tb = new TextBox[nam.Length];
            for(int i=0; i < nam.Length; i++)
            {
                if (i == nam.Length - 1) { if (itk) { break; } }
                iss = false;
                foreach(combo co in com)
                {
                    if (co.id == i)
                    {
                        cb[c] = new ComboBox();
                        cb[c].AutoCompleteSource =
System.Windows.Forms.AutoCompleteSource.ListItems;
                        cb[c].Location = new Point(25, 25 + i * 50);
                        cb[c].Size = new Size(350, 30);
                        foreach(string val in co.values)
                        {
                            cb[c].Items.Add(val);
                        }
                        if (valu != null)
                        {
```

```
                            {
                                Console.WriteLine("here");
                                Console.WriteLine(valu[i]);
                                string curs = "";
                                int stops = 0;
                                for(int m = 0; m < co.values.Count;m++)
                                {
                                    curs = "";
                                    curs = co.values[m];
                                    Console.WriteLine(curs);
                                    for(int k = 0; k < curs.Length; k++)
                                    {
                                        if (curs[k] == '-') { Console.WriteLine(stops);
stops = k;break; }
                                    }
                                    if (valu[i] == curs.Substring(0, stops))
{ cb[c].SelectedItem = curs; }
                                }
                            }
                        else { cb[c].SelectedItem = valu[i]; }
                        }
                        this.Controls.Add(cb[c]);
                        c++;
                        iss = true;
                        break;
                    }
                }
                if (iss) continue;
                tb[t] = new TextBox();
                tb[t].Location = new Point(12, 25 + i * 50);
                tb[t].Size = new Size(350, 30);
                if (valu != null)
                {
                    tb[t].Text = valu[i];
                }
                this.Controls.Add(tb[t]);
                t++;
            }
            InitializeComponent();
            button1.Location = new Point(40, (nam.Length-1) * 50+20);
        }
        private void add_Load(object sender, EventArgs e)
        {
        }

        private void button1_Click(object sender, EventArgs e)
        {
            c = 0;
            t = 0;
            result = "";
            int test1;
            double test2;
            string cur;
            int stop=0;
            bool can_it=true;
            for(int i = 0; i < names.Length; i++)
            {
                iss = false;
                if (i == names.Length - 1) { if (itke) { break; } }
                foreach (combo co in combos)
```

```
                                if (i == 0)
                                {
                                    if(((cb[c].SelectedIndex==-1)&&(cb[c+1].SelectedIndex ==
-1))|| ((cb[c].SelectedIndex != -1) && (cb[c + 1].SelectedIndex != -1)))
                                    {
                                        can_it = false; break;
                                    }
                                    if (cb[c].SelectedIndex == -1)
                                    {
                                        result = result + "NULL"+',';
                                        cur = cb[c + 1].SelectedItem.ToString();
                                        if (forg.Contains(i))
                                        {
                                            for (int k = 0; k < cur.Length; k++)
                                            {
                                                if (cur[k] == '-') { stop = k; break; }
                                            }
                                            cur = cur.Substring(0, stop);
                                        }
                                        result = result + @cur;
                                        if (i != names.Length - 1) { result += ","; }
                                        c=c+2;
                                        i++;
                                        iss = true;
                                        break;
                                    }
                                    if (cb[c+1].SelectedIndex == -1)
                                    {

                                        cur = cb[c].SelectedItem.ToString();
                                        if (forg.Contains(i))
                                        {
                                            for (int k = 0; k < cur.Length; k++)
                                            {
                                                if (cur[k] == '-') { stop = k; break; }
                                            }
                                            cur = cur.Substring(0, stop);
                                        }
                                        result = result + @cur;
                                        if (i != names.Length - 1) { result += ","; }
                                        result = result + "NULL" + ',';
                                        c = c + 2;
                                        i++;
                                        iss = true;
                                        break;
                                    }
                                }
                                else {
                                if (cb[c].SelectedIndex == -1) { can_it = false; break; }
                                cur = cb[c].SelectedItem.ToString();
                                if (forg.Contains(i))
                                {
                                    for (int k = 0; k < cur.Length; k++)
                                    {
                                        if (cur[k] == '-') { stop = k; break; }
                                    }
                                    cur = cur.Substring(0, stop);
                                }
                                result = result + @cur;
```

```
        }
        else
        {


            if (co.id == i)
            {
                if (cb[c].SelectedIndex == -1) { can_it = false; break; }
                cur = cb[c].SelectedItem.ToString();
                if (forg.Contains(i))
                {
                    for (int k = 0; k < cur.Length; k++)
                    {
                        if (cur[k] == '-') { stop = k; break; }
                    }
                    cur = cur.Substring(0, stop);
                }
                result = result + @cur;
                if (i != names.Length - 1) { result += ","; }
                c++;
                iss = true;
                break;
            }
        }

    }
    if (!can_it) { break; }
    if (iss) continue;
    switch (tip[i])
    {
        case "string":
            if (i != names.Length - 1)
            {
                cur = "'" + @tb[t].Text + "',";
            }
            else
            {
                cur = "'" + @tb[t].Text + "'";
            }
            result = result + cur;
            break;
        case "int":
            can_it= int.TryParse(tb[t].Text, out test1);
            if (can_it) {
            if (i != names.Length - 1)
            {
                result = result + @tb[t].Text + ",";
            }
            else
            {
                result = result + @tb[t].Text;
            }

            }
            break;
        case "double":
            cur = @tb[t].Text;
            cur = cur.Replace(".", ",");
            can_it = double.TryParse(cur, out test2);
```

```
                        }
                        else
                        {
                            result = result + @cur;
                        }
                    }
                    break;

                }
                if (!can_it) { break; }
                t++;
            }
            if (can_it) { this.DialogResult = DialogResult.OK; }
            else { MessageBox.Show("Ошибка");}
        }
    }
}
```

form3.cs

```
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;
using Microsoft.Data.SqlClient;
namespace WindowsFormsApp3
{
    public partial class Form4 : Form
    {
        string conect;
        public Form4(string con)
        {
            con = conect;
            InitializeComponent();
        }
        private void button1_Click_1(object sender, EventArgs e)
        {
            Standart_table table1 = new Standart_table();
            table1.sqlcon = $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated
Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            table1.sqlq = "SELECT * FROM case_func2()";
            table1.names = new string[] { "Тип договора", "Цена", "Адрес", "Квартира",
"Название фирмы", "Наценка", "Тип фирмы" };
            Form5 form = new Form5(table1);
            form.Show();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            Standart_table table1 = new Standart_table();
            table1.sqlcon = $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated
Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            table1.sqlq = "SELECT * FROM testfunction3()";
            table1.names = new string[] { "Тип договора", "Цена", "Адрес",
"Название фирмы", "Наценка", "Тип фирмы" };
```

```
        private void button3_Click_1(object sender, EventArgs e)
        {
            Standart_table table1 = new Standart_table();
            table1.sqlcon = $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated
Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            table1.sqlq = "SELECT f.name,(SELECT sum(d.cost) FROM dogovor d WHERE
d.firma_id=f.id AND d.type='Аренда')AS" + '"' + "Сумма аренды" + '"' +
",(SELECT
count(d.cost) FROM dogovor d WHERE d.firma_id=f.id AND d.type='Аренда')AS" + '"' + "Кол -
во аренды" + '"' + ",(SELECT sum(d.cost) FROM dogovor d WHERE d.firma_id=f.id AND
d.type='Покупка')AS" + '"' + "Сумма покупок" + '"' + ",(SELECT count(d.cost) FROM dogovor
d WHERE d.firma_id=f.id AND d.type='Покупка')AS " + '"' + "Кол - во покупок" + '"' +
" ,doga_avg AS " + '"' + "Общее кол-во" + '"' + " FROM firma f,(SELECT dd.firma_id
AS  fid, COUNT(*) AS doga_avg FROM dogovor dd GROUP BY dd.firma_id )as avg_dog WHERE
avg_dog.fid=f.id AND(0 < ALL(SELECT COUNT(*) FROM dogovor d WHERE d.firma_id=f.id AND
d.type='Аренда'))AND(1 < ALL(SELECT COUNT(*) FROM dogovor d WHERE d.firma_id=f.id AND
d.type='Покупка'))";
            table1.names = new string[] { "Название фирмы", "Сумма аренды", "Кол-во
аренды", "Сумма покупок", "Кол-во покупок", "Общее кол-во"
            };  Form5 form = new Form5(table1);
            form.Show();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Standart_table table1 = new Standart_table();
            table1.sqlcon = $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated
Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            table1.sqlq = "SELECT uchastok_id,floor,flat,square,cost,type,number FROM
nedvishimost pp WHERE pp.square = (SELECT max(ppm.square) FROM nedvishimost ppm
WHERE  ppm.uchastok_id = pp.uchastok_id)";
            table1.names = new string[] { "id Участка", "Кол-во этажей",
"Площадь",  "Кадастровая стоимость", "Тип", "Кадастровый номер" };
            Form5 form = new Form5(table1);
            form.Show();
        }

        private void button5_Click_1(object sender, EventArgs e)
        {
            string sqlcon = $@"Data Source=.\SQLEXPRESS;Initial Catalog=reestr;Integrated
Security=False;User
Id=Dbadmin;Password=2803;MultipleActiveResultSets=True;TrustServerCertificate=true;";
            string sqlq = "EXECUTE trig";
            using (SqlConnection connection = new SqlConnection(sqlcon))
            {
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.CommandText = sqlq;
                command.Connection = connection;
                command.ExecuteNonQuery();
                connection.Close();
            }
        }
    }
}
```

form4.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Windows.Forms;
using Microsoft.Data.SqlClient;
namespace WindowsFormsApp3
{
    public partial class Form5 : Form
    {
        Standart_table table;
        public Form5(Standart_table tables)
        {
            table = tables;
            InitializeComponent();
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            dataGridView1.AllowUserToAddRows = false;
            dataGridView1.AllowUserToDeleteRows = false;
            dataGridView1.ReadOnly = true;
            dataGridView1.AllowUserToOrderColumns = false;
            DataGridViewTextBoxColumn[] dgvc;
            dgvc = new DataGridViewTextBoxColumn[table.names.Length];
            for (int i = 0; i < table.names.Length; i++)
            {
                dgvc[i] = new DataGridViewTextBoxColumn();
                dgvc[i].HeaderText = table.names[i];
                dgvc[i].Name = "col" + i;
                dgvc[i].SortMode = DataGridViewColumnSortMode.NotSortable;
                dataGridView1.Columns.Add(dgvc[i]);
            }
            using (SqlConnection connection = new SqlConnection(table.sqlcon))
            {
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.CommandText = table.sqlq;
                command.Connection = connection;
                SqlDataReader reader = command.ExecuteReader();
                int count = reader.FieldCount;
                string[] s; s = new string[reader.FieldCount];
                while (reader.Read())
                {
                    for (int i = 0; i < count; i++)
                    {
                        s[i] = String.Format("{0}", reader[i]);
                    }
                    dataGridView1.Rows.Add(s);
                }
                connection.Close();
            }
        }
    }
}
```

form5.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp3
{
```

```
        public combo(int i, List<string> v)
        {
            id = i;
            values = v;
        }
    }
    public class constr
    {
        public int id;
        public string tip;
        public constr(int i, string c)
        {
            id = i;
            tip = c;
        }
    }
    public class foreign
    {
        public int id;
        public string fsqlq;
        public foreign(int i, string c)
        {
            id = i;
            fsqlq = c;
        }
    }
    public class Standart_table
    {
        public int unique=-1;
        public string tip;
        public string[] names;
        public List<foreign> foreigns;
        public List<constr> constrains;
        public List<combo> combos;
        public string sqlcon;
        public string sqlq;
        public int[] shirina;
        public string table;
    }
}
```

Standart_table.cs