Учебное пособие для практических задач

Практический Курс

Дмитрий Иванов

Создание Windows® приложений для работы с Базами Данных

Простыми словами

2017

1. Введение.

1.1 Введение.

Давайте представим, что вы ведете учет чего-либо в обычной Excel таблице, как показано на рисунке 1.

Учет проката техники						
Имя клиента	**Телефон**	**Название техники**	**Цена за день**	**Дата начала**	**Дата конца**	**Стоимость**
KH Services	7 495 555-55-55	Бульдозер ТМ-10	12000	01.05.2016	03.05.2016	36000
euroasfalt	7 925 455-66-77	Каток ДУ-50	8500	02.05.2016	05.05.2016	34000
KH Services	7 495 555-55-55	Автокран К-4561	6500	02.05.2016	03.05.2016	13000
Главстрой	7 495 777-77-77	9.5т, Тент, Man	14000	06.05.2016	06.05.2016	14000
Мастер Бетона	7 499 844-44-44	Kenworth W900	15500	07.05.2016	08.05.2016	31000
euroasfalt	7 925 455-66-77	Каток ДУ-50	8500	09.05.2016	09.05.2016	8500
KH Services	7 495 555-55-55	Kenworth W900	12000	07.05.2016	13.05.2016	84000

Рис.1

И казалось бы, этого вполне достаточно, для полноценного учета. Но, к сожалению, такой подход имеет ряд проблем, которые мы сейчас разберем.

Допустим, вы предприниматель, предоставляющий строительную технику в аренду другим компаниям и ведете учет в простой табличке, как уже было показано на рис.1

Проблема 1.

В первую очередь обратите внимание, что произойдет, если у какой-либо компании изменится номер телефона? Допустим, что компания "KH Services" изменила телефон на 7 495 555-65-65. Чтобы содержать таблицу актуальной, необходимо сделать изменения в трех строках (а если их уже 500?). Если мы не сделаем этого, наши данные будут противоречивыми, впоследствии мы не сможем определить, какой из телефонных номеров является актуальным.

Проблема 2.

Мы можем сделать опечатку в названии компании и случайно записать KJ Service вместо KH Service. Впоследствии будет не ясно, либо это новый клиент, либо была сделана опечатка. Такая ошибка не могла бы возникнуть, если бы мы выбирали компанию из списка.

Проблема 3.

У вас не получиться вести учет совместно с несколькими людьми.

Например, вашему бухгалтеру необходим учет счетов и платежей, а сотрудники отдела проката хотят вести учет заказов и встреч с клиентами, но например бухгалтерия не хотела бы, чтобы сотрудники отдела проката имели доступ к данным о платежах.

Более серьезные проблемы начинаются, когда приходит время проанализировать накопившиеся данные, например:

- подсчитать доход за необходимый промежуток времени.
- подсчитать количество арендованной техники определенной фирмой.
- узнать какая техника в настоящий момент свободна и т.д.

И когда в вашей таблице накопится много информации, подобные проблемы обязательно возникнут, а также и другие в зависимости от специфики вашей учетной деятельности.

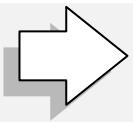
1.2 База данных

Теперь, когда нам ясны проблемы ведения учета в одной таблице, разберемся, как же организовать все правильно.

Для полноценного ведения учета необходимо использовать так называемую «Базу данных».

Что такое база данных? Вы можете представить ее себе как несколько Excel таблиц, которые связаны между собой особым образом. Также каждая таблица создается, под конкретную сущность, но об этом более подробно в следующей главе.

1.3 Выявление сущностей



Сущность - это важная вещь или объект, сведения о котором нужно сохранить.

Как вы видите, в таблице на рис.1, все свалено в кучу, все сущности, такие как:

- клиент (столбец: "Имя клиента", "Телефон")

- арендуемая техника (столбец: "Название техники", "Цена за день")

- прокат (столбец: "Дата начала", "Дата конца", "Стоимость")

ВАЖНО!

Перечисленные выше сущности являются самостоятельными и каждая такая сущность должна иметь свою собственную таблицу. Этот процесс также называется нормализацией.

Как результат, вместо одной таблицы, у нас будет 3, как показано на рис.2

Client

Имя клиента	Телефон
KH Services	7 495 555-55-55
euroasfalt	7 925 455-66-77
Главстрой	7 495 777-77-77
Мастер Бетона	7 499 844-44-44

Equipment

Название техники	Цена за день
Бульдозер ТМ-10	12000
Каток ДУ-50	8500
Автокран К-4561	6500
9.5т, Тент, Man	14000
Kenworth W900	15500

Rent

Дата начала	Дата конца	Стоимость
01.05.2016	03.05.2016	36000
02.05.2016	05.05.2016	34000
02.05.2016	03.05.2016	13000
06.05.2016	06.05.2016	14000
07.05.2016	08.05.2016	31000
09.05.2016	09.05.2016	8500
07.05.2016	13.05.2016	84000

Рис.2

Таблица **Client** – содержит всех наших клиентов и их телефонные номера.

Таблица **Equipment** – содержит наименования техники, которой мы располагаем, а также ее стоимость за 1 день аренды.

Таблица **Rent** – содержит информацию о сроках арендованной техники.

Теперь описанные выше проблемы в главе 1.1 решены:

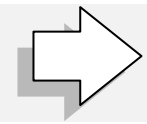
1. Когда у компании сменится телефонный номер, его нужно будет изменить только в одной строке, в таблице с именем **Client**

2. Нет необходимости каждый раз вручную писать название компании, избежав, таким образом опечатки. Мы просто выберем необходимую компанию из списка.

3. Разные пользователи базы данных могут иметь доступ только к определенным таблицам, например только менеджер может добавить новую компанию в качестве клиента в таблицу **Client**

1.4 Связывание таблиц и внешние ключи

Как вы могли заметить, теперь у нас 3 отдельные таблицы, которые между собой никак не связаны и глядя на них совершенно не ясно, какую технику взяла компания в аренду и на какие сроки. А чтобы стало ясно, в таблицу необходимо добавить связи.



Связи между таблицами, пожалуй, одна из самых важных тем в базах данных.

После того, как мы добавим необходимые связи, наши таблицы будут выглядеть так, как показано на рис.3

Client

id	Имя клиента	Телефон
1	KH Services	7 495 555-55-55
2	euroasfalt	7 925 455-66-77
3	Главстрой	7 495 777-77-77
4	Мастер Бетона	7 499 844-44-44

Equipment

id	Название техники	Цена за день
1	Бульдозер ТМ-10	12000
2	Каток ДУ-50	8500
3	Автокран К-4561	6500
4	9.5т, Тент, Man	14000
5	Kenworth W900	15500

Rent

id	id_Client	id_Equipment	Дата начала	Дата конца	Стоимость
1	1	1	01.05.2016	03.05.2016	36000
2	2	2	02.05.2016	05.05.2016	34000
3	1	3	02.05.2016	03.05.2016	13000
4	3	4	06.05.2016	06.05.2016	14000
5	4	5	07.05.2016	08.05.2016	31000
6	2	1	09.05.2016	09.05.2016	8500
7	1	5	07.05.2016	13.05.2016	84000

Рис.3

Давайте подробней разберемся со связями, глядя на рис.3

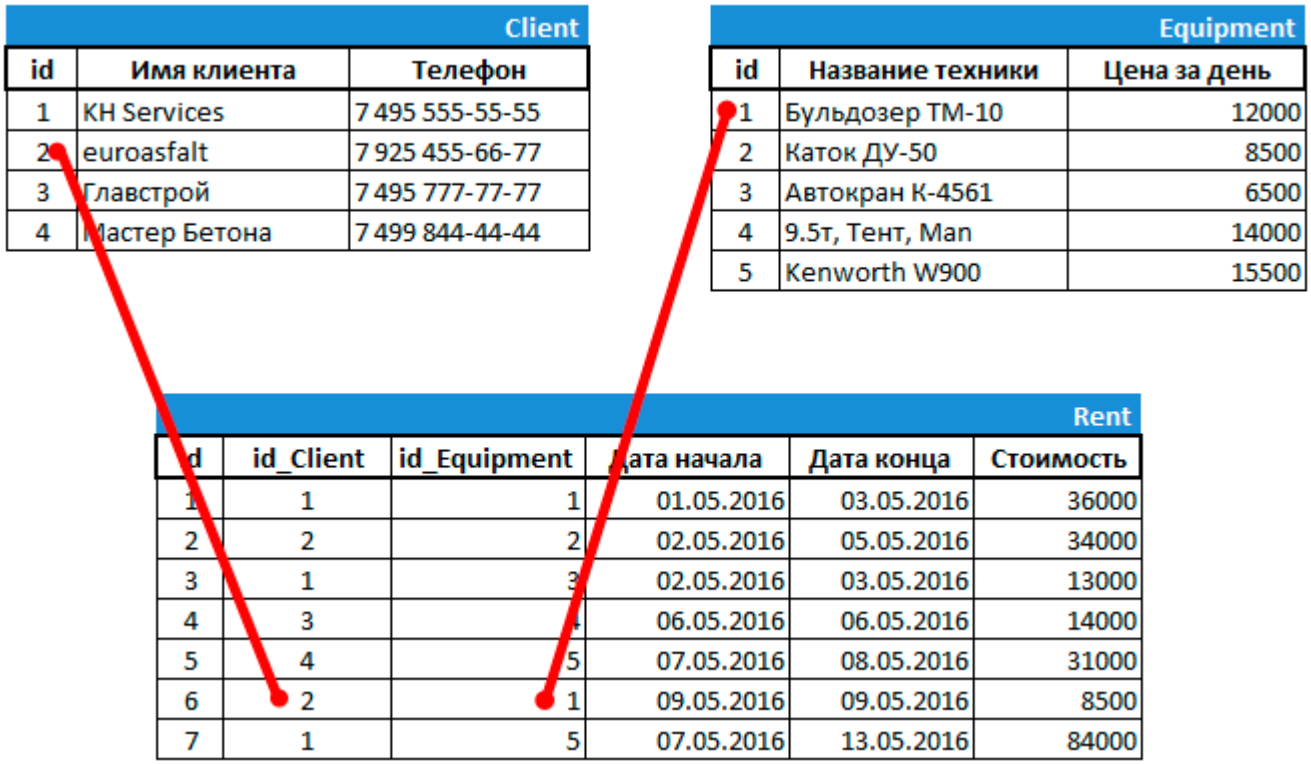
Обратите внимание, теперь каждая таблица имеет столбец с именем **id**. Благодаря этому столбцу, каждая запись в таблице имеет уникальный идентификатор, правильно называть его «**Первичный ключ**». При добавлении новой записи в таблицу, ей автоматически присвоится уникальный идентификатор.

В таблице **Rent** вы также можете заметить появление столбцов **id_Client** и **id_Equipment**, это «**внешние ключи**».

Именно с помощью **Первичных ключей** и **Внешних ключей** образуются связи между таблицами.

Далее все просто, внешний ключ **id_Client** таблицы **Rent** указывает на первичный ключ таблицы **Client.** Таким образом, мы знаем, к какому клиенту относиться каждая запись в таблице **Rent**.

Чтобы было совсем наглядно, взгляните на рис.4



Client

id	Имя клиента	Телефон
1	KH Services	7 495 555-55-55
2	euroasfalt	7 925 455-66-77
3	Главстрой	7 495 777-77-77
4	Мастер Бетона	7 499 844-44-44

Equipment

id	Название техники	Цена за день
1	Бульдозер ТМ-10	12000
2	Каток ДУ-50	8500
3	Автокран К-4561	6500
4	9.5т, Тент, Man	14000
5	Kenworth W900	15500

Rent

id	id_Client	id_Equipment	Дата начала	Дата конца	Стоимость
1	1	1	01.05.2016	03.05.2016	36000
2	2	2	02.05.2016	05.05.2016	34000
3	1	3	02.05.2016	03.05.2016	13000
4	3	4	06.05.2016	06.05.2016	14000
5	4	5	07.05.2016	08.05.2016	31000
6	2	1	09.05.2016	09.05.2016	8500
7	1	5	07.05.2016	13.05.2016	84000

Рис.4

Т.е. для записи в таблице **Rent** с идентификатором **6**, соответствует клиент «euroasfalt», который взял в аренду технику «Бульдозер ТМ-10».

1.5 Типы данных

Прежде чем приступить к созданию первого приложения для работы с базой данных, нам осталось рассмотреть еще одну тему. Речь идет о типах данных используемых в столбцах.

На первый взгляд может показаться, что все данные в таблицах это просто текст, но это не так. Каждый столбец в таблице соответствует, какому либо типу данных, перечислим их:

- **текстовый** (просто любой текст, например фамилия или название компании)
- **целочисленный** (без дробной части, например количество чего либо)
- **число с плавающей запятой** (например число 3,14)
- **денежный** (например 25.00 руб)
- **логический** (принимает значение Да или Нет)
- **дата** (например 02.06.2016)
- **время** (например 19:57:00)
- **дата и время** (например 02.06.2016 19:57:00)
- **файл** (позволяет сохранить файл непосредственно в базе данных)
- **изображение** (позволяет сохранить фотографию в базе данных)
- **внешний ключ** (о нем мы говорили в разделе 1.3)

Поэтому когда вы будете создавать столбцы в таблице, вам необходимо выбрать его тип. Зачем они нужны? Это помогает более рационально хранить данные и обеспечивает более высокую работу базы данных. Также это позволяет избежать ошибок. Допустим, если столбец принадлежит целочисленному типу, пользователь не сможет сохранить там какой либо текст.

2. Практика.

2.1 Создание первого приложения для работы с БД.

Теперь мы знаем достаточно теории о базах данных, чтобы сделать свое собственное приложение для работы с ней. Более опытный читатель, возможно, вспомнит о языке SQL запросов, о котором еще не было сказано ни слова, но к счастью современные инструменты разработки позволяют первое время обойтись и без него. Но если вы решите более серьезно заняться базами данных, то несомненно вам продеться ознакомиться с SQL.

Для наших практических задач я остановил свой выбор на использовании довольно простой среды разработки баз данных, в которой собрано все, для того чтобы немедленно приступить к созданию структуры базы данных и приложению, которое будет работать с ней.

Посетите сайт http://myvisualdatabase.com/ для получения последней версии My Visual Database.

Программа, к сожалению, не бесплатная, но имеет ознакомительный период, которого нам будет достаточна для учебных целей. Но можно найти более раннюю бесплатную версию по данной ссылке: http://myvisualdatabase.com/download/myvisualdb1.44.exe

Кратко о возможностях программы:

- Создание структуры БД
- Создание форм для работы с БД
- Отчетная система для печати и анализа
- Скриптовый язык, позволяющий реализовать любой нестандартный функционал
- Поддержка СУБД SQLite и MySQL
- Возможность создания простого web доступа к БД
- Создание полноценного Windows приложения

После установки My Visual Database она немедленно готова к работе, дополнительная настройка не требуется (рис.5)

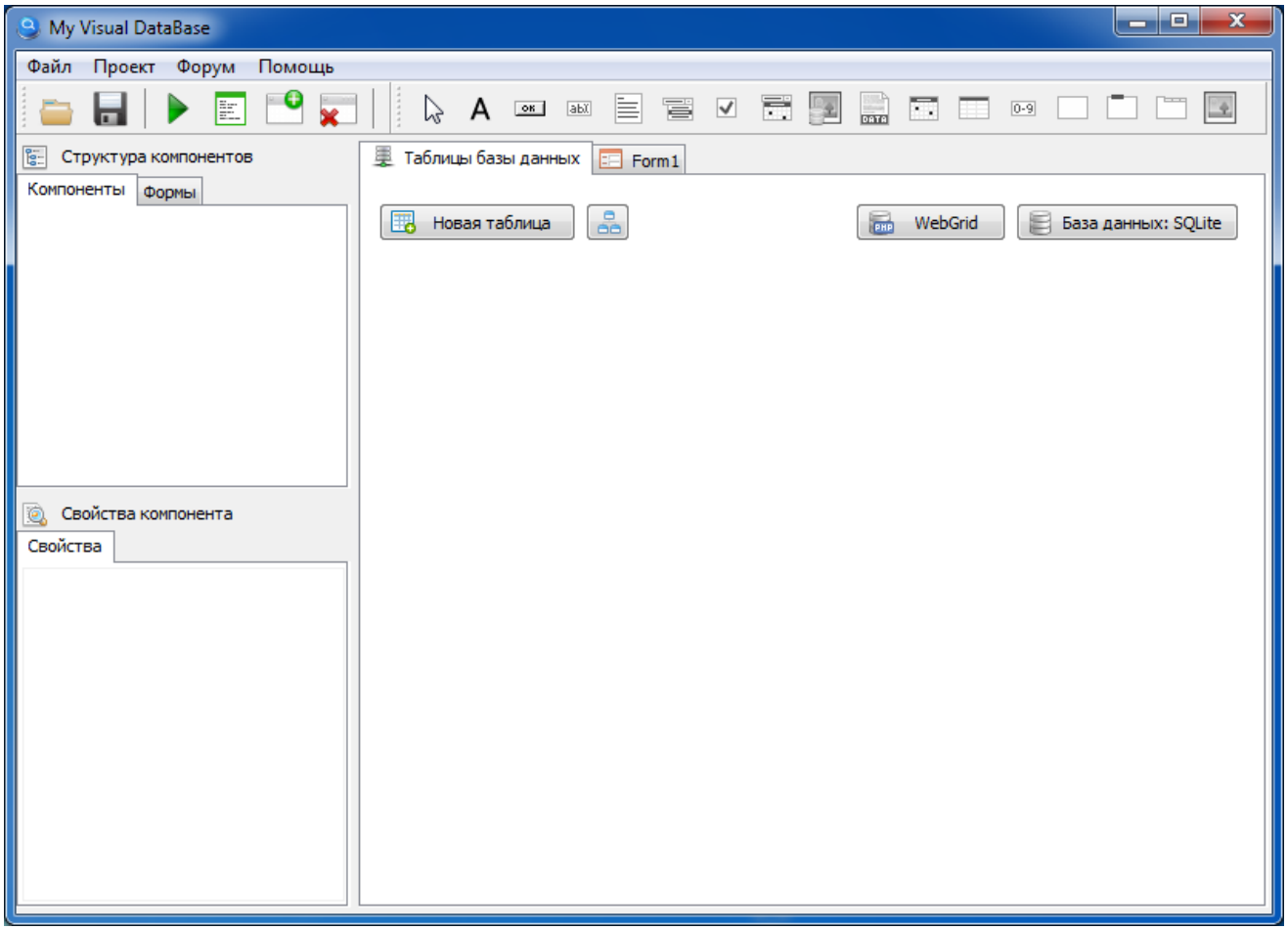


Рис.5 Внешний вид программы My Visual Database

Теперь мы можем приступить к созданию нашего первого приложения. В качестве примера создадим уже рассмотренный выше учет строительной техники.

Прежде чем приступить к созданию приложения, сохраните проект в отдельную папку (меню: Файл > Сохранить как…), назовем наш проект например *Rent* (рис.6)

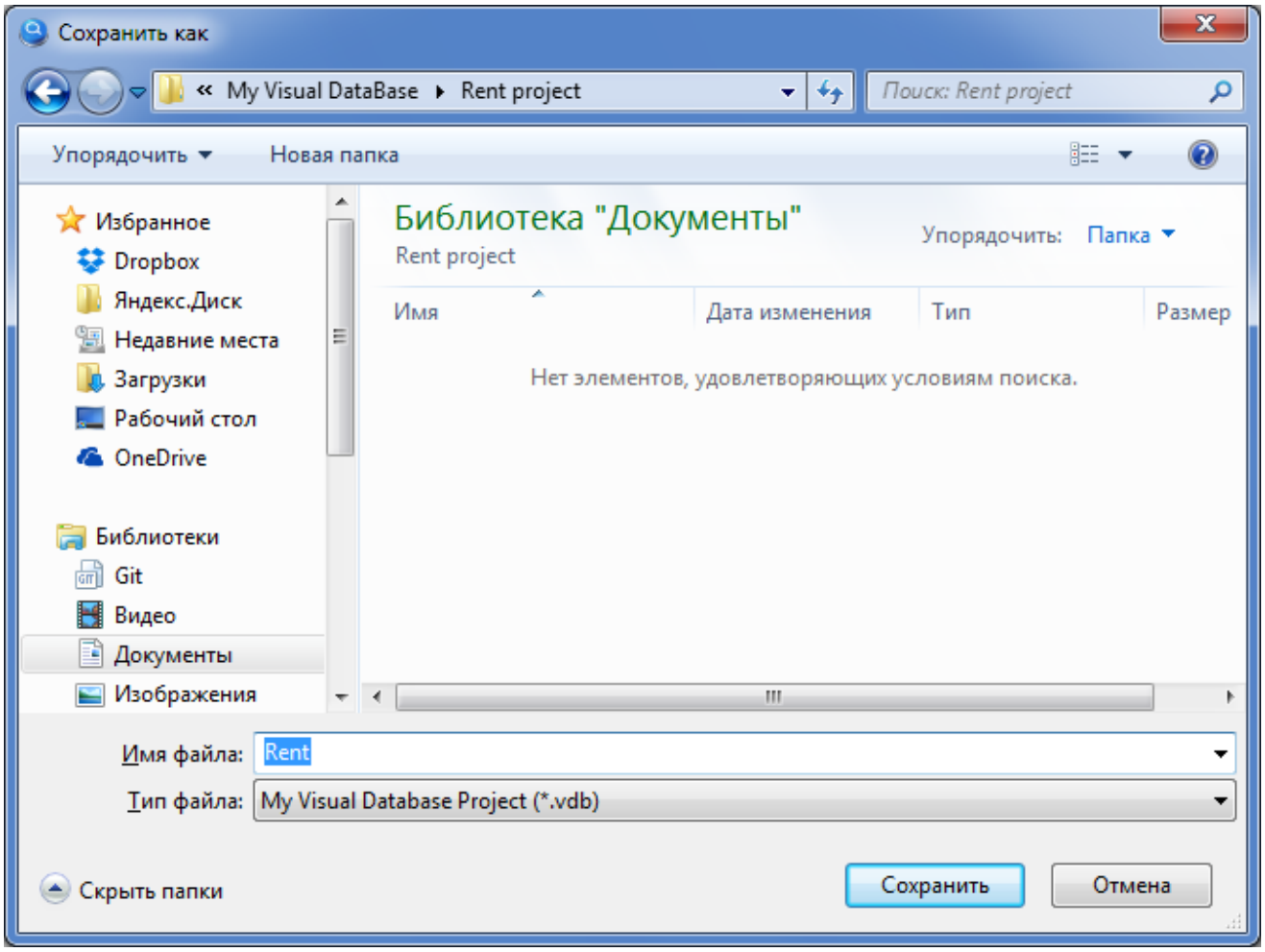


Рис.6 Сохранение проекта

Как мы помним из первой главы книги, база данных состоит из таблиц и столбцов, именно их нам сейчас и предстоит создать.

Начнем с создания таблиц:

- Client
- Equipment
- Rent

На вкладке «Таблицы базы данных» нажмите кнопку **«Новая таблица»** чтобы создать нашу первую таблицу *Client,* таким же образом создайте таблицы *Equipment* и *Rent*

После создания таблиц, у нас должно получиться, как показано на рис. 7

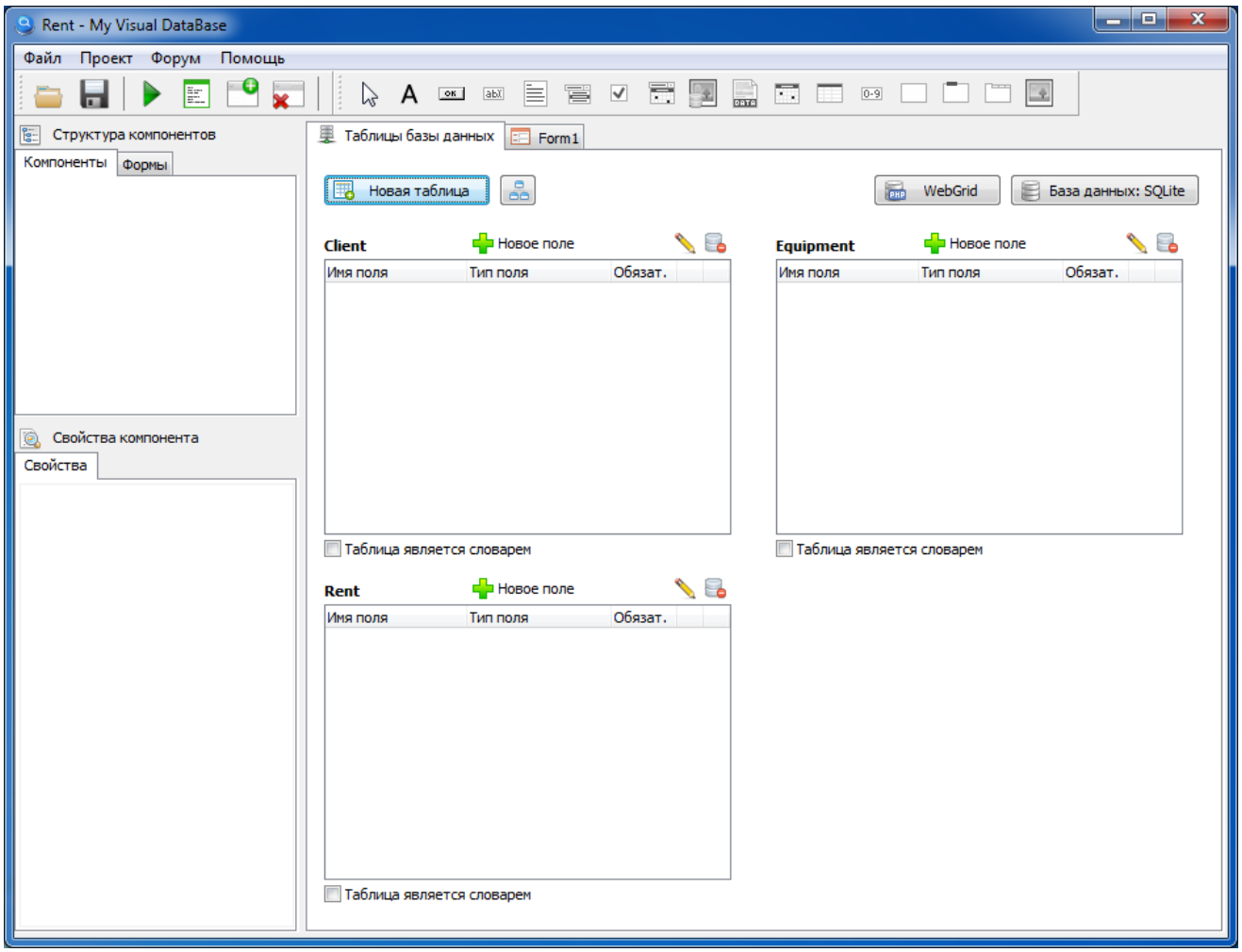


Рис.7

Снова обратите внимание на рисунок 7, у каждой созданной таблицы сверху есть кнопка «Новое поле», используя эту кнопку необходимо создать поля для таблиц. Давайте создадим первое поле. Нажмите кнопку «Новое поле» в таблице **Client**, затем из появившегося меню выберите тип **ТЕКСТ** и введите название поля **Name**. Таким же образом создайте все перечисленные ниже поля для остальных таблиц. Не забудьте правильно выбрать тип поля.

Таблица **Client**:

- Name (имя клиента, тип ТЕКСТ)
- Phone (телефон клиента, тип ТЕКСТ)

Таблица **Equipment**

- Tech (название техники, тип ТЕКСТ)
- CostPerDay (цена за 1 день, тип ДЕНЬГИ)

Таблица **Rent**

- DateStart (дата начала аренды, тип ДАТА)
- DateEnd (дата конца аренды, тип ДАТА)
- TotalCost (общая стоимость, тип ДЕНЬГИ)

Результат вашей работы должен быть таким, как показано на рис.8

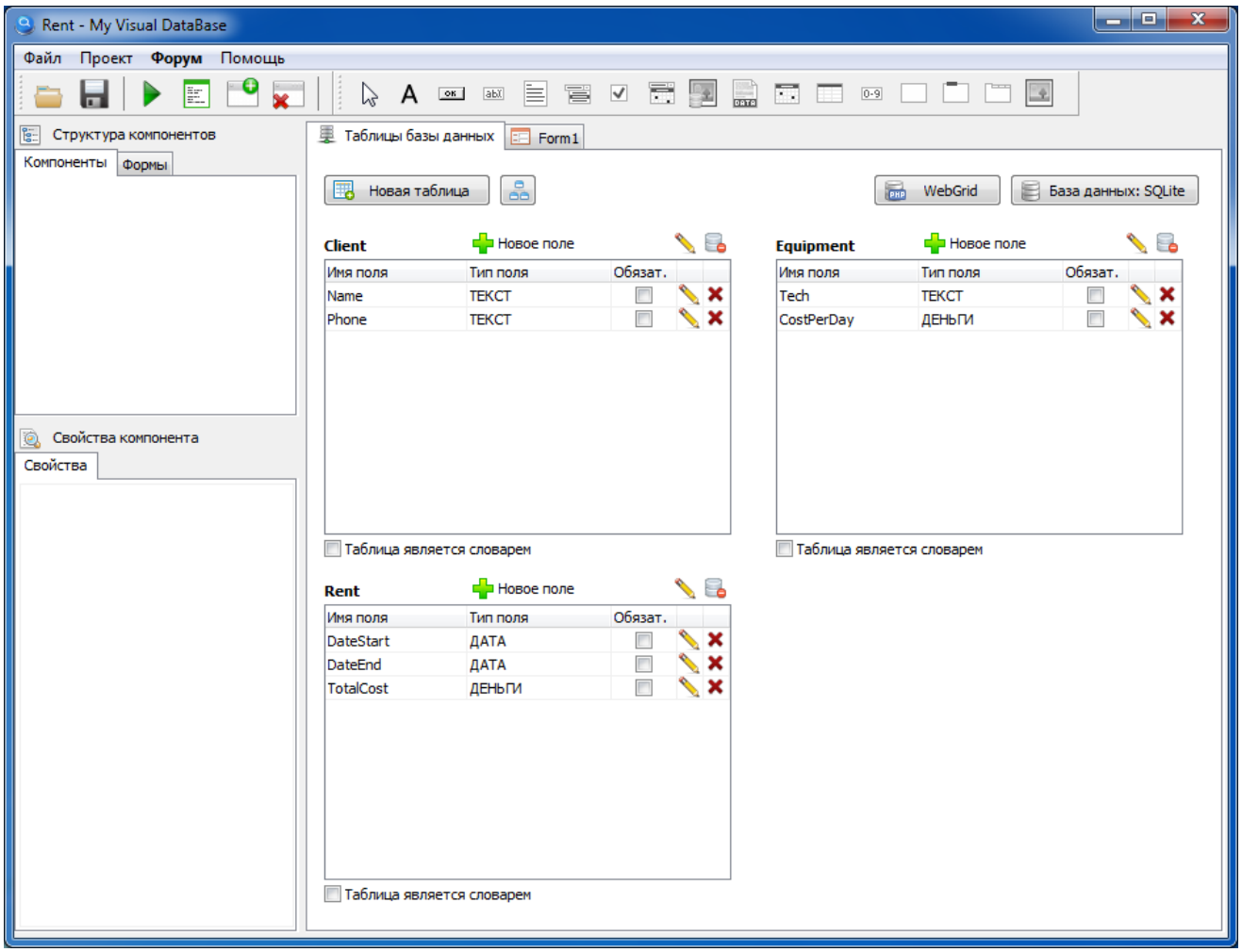


Рис.8

Возможно, вы помните, что каждая таблица должна содержать поле с именем id, в котором хранится уникальный идентификатор для каждой записи. Вам не нужно беспокоиться об этом, программа создает его автоматически, но не показывает данное поле в таблицах.

Структура БД почти готова, осталось создать только два внешних ключа в таблице **Rent**, которые будут связываться с таблицами **Client** и **Equipment**. Это необходимо, чтобы, когда мы будем создавать новую запись об аренде, мы смогли бы выбрать клиента и арендованную им технику из списка и этот выбор будет сохранен именно во внешних ключах.

Создание внешних ключей происходит почти также как и создание полей. В таблице **Rent** нажмите кнопку **«Новое поле»** и из появившегося меню выберите пункт **«Связь»**, затем выберите из списка таблицу **Client** и нажмите **«ОК»** как показано на рис. 9

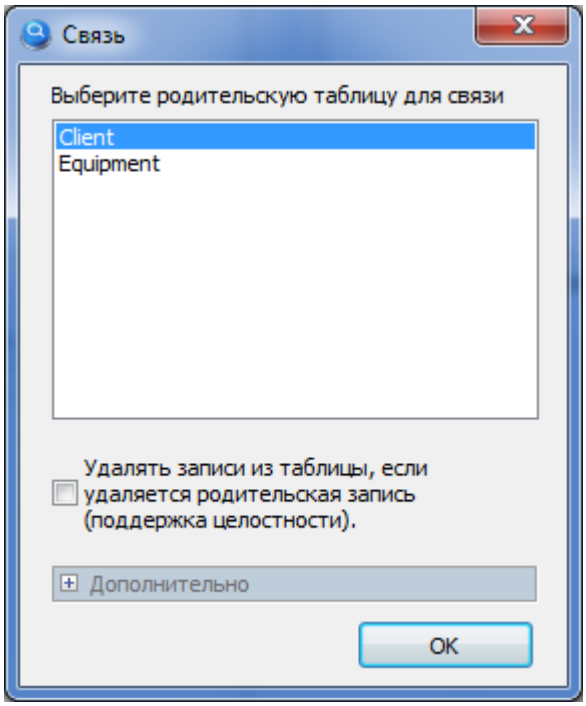


Рис.9

Тоже самое проделайте для создания внешнего ключа на таблицу **Equipment**. Опять в таблице **Rent** нажмите кнопку **«Новое поле»** и из появившегося меню выберите пункт **«Связь»**, затем выберите из списка теперь уже таблицу **Equipment** и нажмите **«ОК»**

Окончательный вид таблиц и полей показан на рис.10

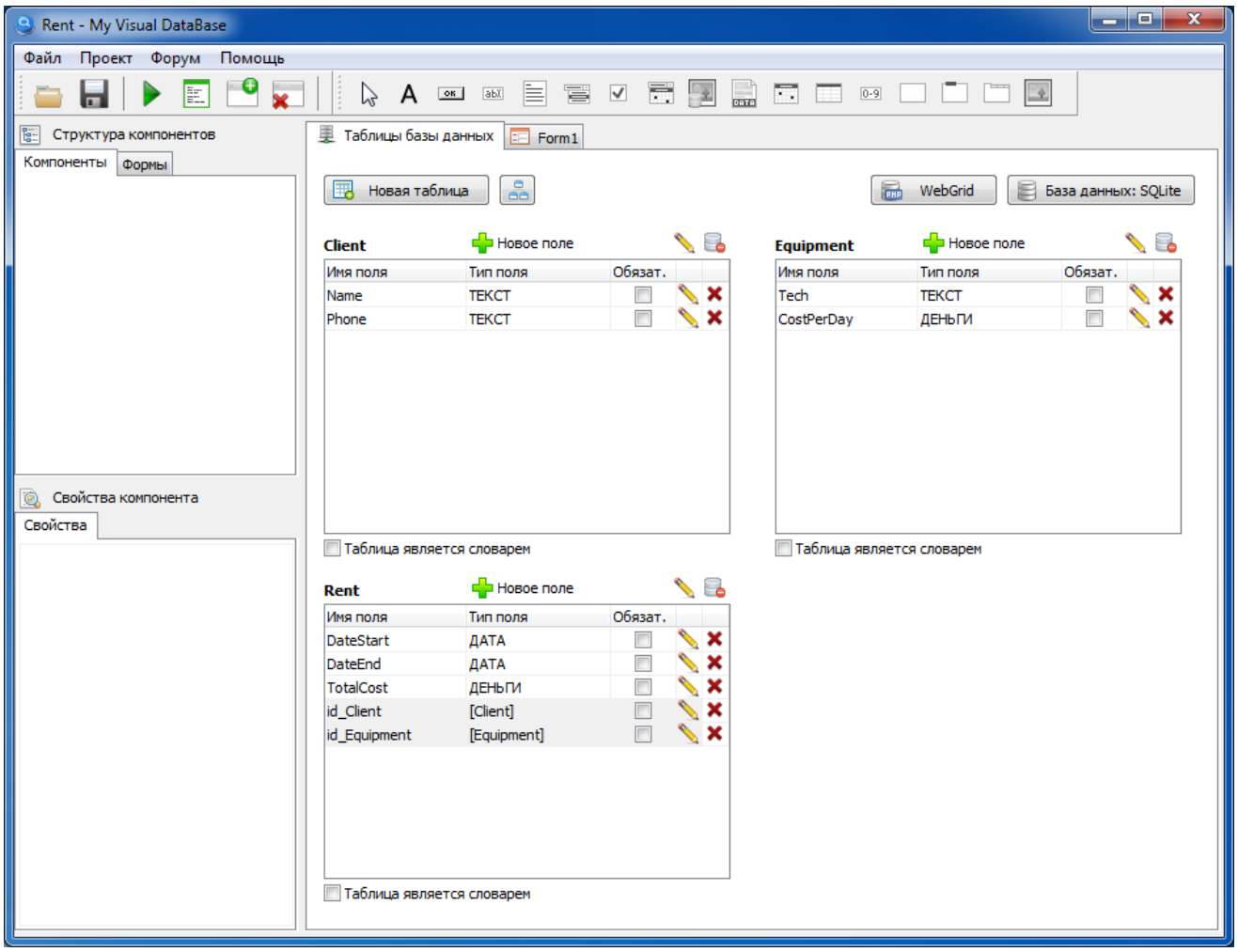
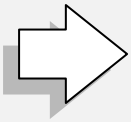


Рис.10

На этом наша структура базы данных полностью готова.

Далее нам необходимо создать пользовательский интерфейс для взаимодействия с базой данных, чтобы мы могли создавать, редактировать и удалять записи. Также необходимо чтобы приложение умело искать записи по различным критериям.

Уже сейчас вы можете запустить ваше приложение, нажав кнопку  на панели инструментов и перед вами появится окно Form1, на котором пока ничего нет.



После первого же запуска приложения, в папке проекта вы найдете exe файл, т.о. ваш проект становится самостоятельным приложением, которое готово работать на любом компьютере без установки каких либо дополнительных компонентов.

Давайте определимся, какие формы нам необходимы.

1. **Form1**
   Форма для поиска. Эта же форма будет главной, именно она будет появляться при запуске вашего приложения.

2. **frmRent**
   Форма для создания и редактирования записи об аренде. Данная форма будет записывать данные в таблицу БД Rent.

3. **frmClientList**
   Форма со списком клиентов.

4. **frmClient**
   Форма создания и редактирования клиента. Данная форма будет записывать данные в таблицу БД Client.

5. **frmTechList**
   Форма со списком строительной техники.

6. **frmTech**
   Форма создания и редактирования информации о строительной техники. Данная форма будет записывать данные в таблицу БД Equipment.

Как правило, для каждой таблицы БД необходимо две формы, одна для вывода всех записей из нее и вторая для создания или редактирования записей для данной таблицы.

Для создания интерфейса вам доступны компоненты, которые вы можете видеть на панели инструментов (рис.11)

Рис.11

Почти все компоненты стандартные и, как правило, применяются в большинстве Windows приложений. Но на всякий случай перечислим их и дадим краткое описание:

- надпись (Label). Позволяет поместить надпись на форме. Как правило, компонент не несет какой либо функциональности.

- кнопка (Button). Важный и часто используемый компонент. Например, может служить для сохранения записи в базу данных.

- текстовое поле (TextBox). Служит для ввода числовой и текстовой информации.

- многострочное текстовое поле (Memo). Также служит для ввода текстовой информации.

- выпадающий список (ComboBox). Позволяет выбрать какое либо значение из списка.

- флажок (CheckBox). Имеет два состояние, установлен либо сброшен.

- дата/время (DateTimePicker). Служит для ввода даты или времени.

- рисунок БД (DBImage). Позволяет сохранить рисунок в базу данных.

- файл БД (DBFile). Позволяет сохранить файл в базу данных.

- календарь (Calendar). Служит исключительно для выбора даты.

- таблица (TableGrid). Позволяет вывести записи из базы данных в табличном виде.

- счетчик (Counter). Позволяет присвоить записи уникальный номер.

- панель (Panel). Декоративный элемент интерфейса.

- группа (Group). Декоративный элемент интерфейса.

- вкладки (PageControl). Позволяет создавать переключаемые вкладки, на каждой вкладке можно разместить другие компоненты.

- рисунок (Image). Позволяет разместить на форме любой рисунок.

Приступим к созданию первой формы, которая будет служить нам для поиска записей. Также на форме будут размещены кнопки, позволяющие вызвать форму для создания или редактирования записи.

Для создания формы будет достаточно следующих компонентов:

Перейдите на вкладку Form1 как показано на рисунке 11а.

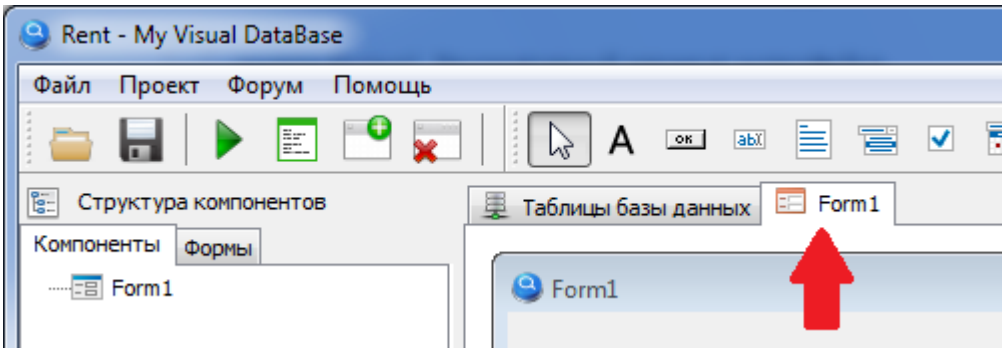


Рис.11а

Выбрав нужный компонент на панели, нажмите левой кнопкой мыши на форме Form1 в том месте, где желаете поместить данный компонент.

Разместив на форме компонент A или OK, вам необходимо изменить текст на нем, сделать это можно с помощью свойства **Caption** на панели **«Свойства компонента»** (рис.12)

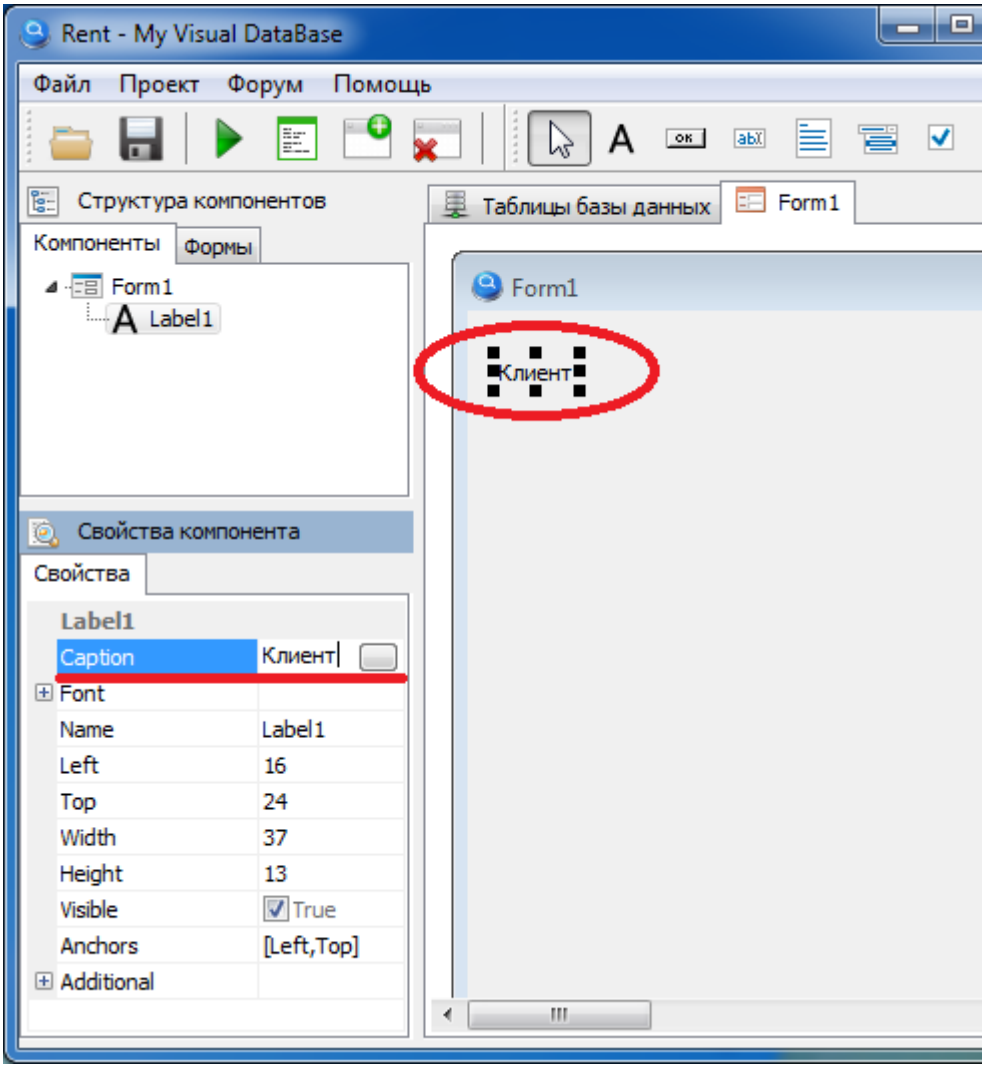


Рис.12

Поместите все компоненты на форму Form1 как показано на рисунке 13. Для вашего удобства цветными линиями обозначены, куда и какие именно компоненты необходимо расположить на форме.

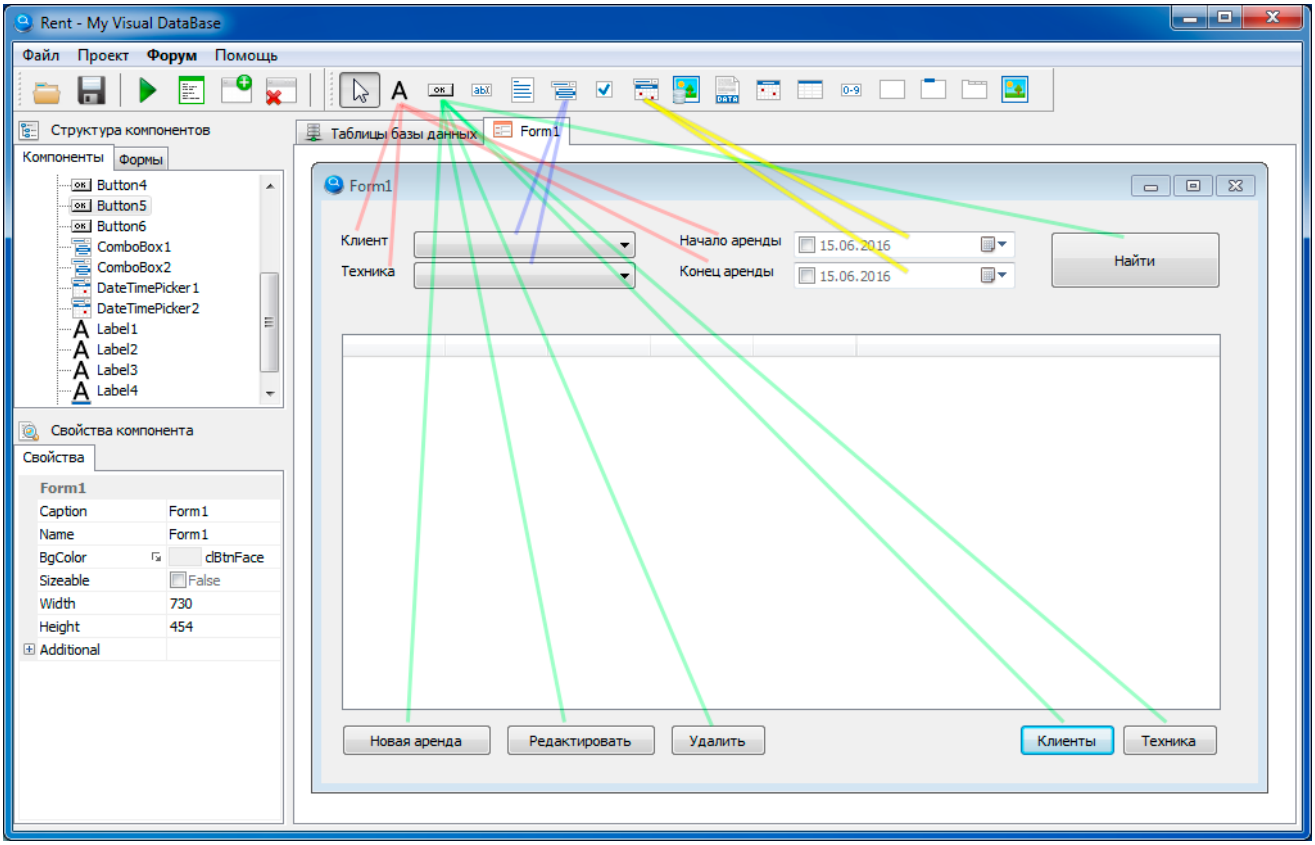


Рис.13

Форма для поиска готова. На данной форме вы сможете найти все записи об аренде конкретного клиента, либо конкретной техники, также найти записи по дате начала либо конца аренды. Вызвать форму для создания записи о новой аренде, редактирования существующей либо удалить запись об аренде.

Также на форме есть кнопки «Клиенты» и «Техника», данные кнопки необходимы для вызова формы со списком всех клиентов и техники соответственно.

Ради любопытства можете нажать кнопку  , чтобы снова запустить наш проект и посмотреть, как он будет выглядеть. Так как пока компоненты на форме не настроены, при нажатии на них, ничего не произойдет.

Приступим к созданию второй формы, которая необходима для создания либо редактирования записи об аренде. Напомню, как правило, используется одна и та же форма, как для создания, так и для редактирования записи.

Создайте вторую форму, нажав кнопку  . Введите название формы **frmRent**

Поместите компоненты на форму **frmRent**, как показано на рисунке 14.

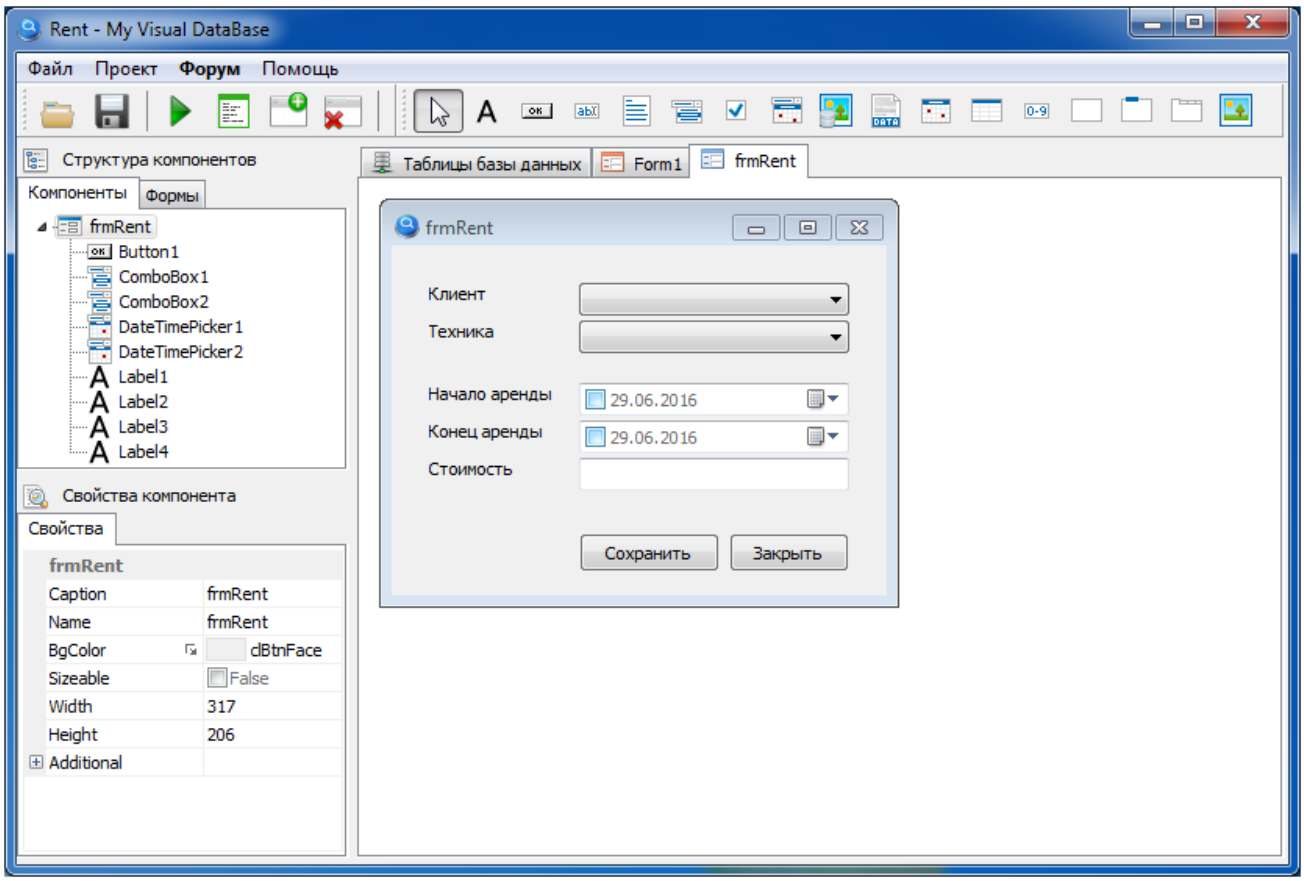


Рис.14

Создадим третью форму с именем **frmClientList,** нажав кнопку  .

На создаваемой форме мы сможем видеть всех наших клиентов, а так же вызвать форму для создания нового клиента либо редактирования существующего.

Поместите компоненты на данную форму как показано на рисунке 15.

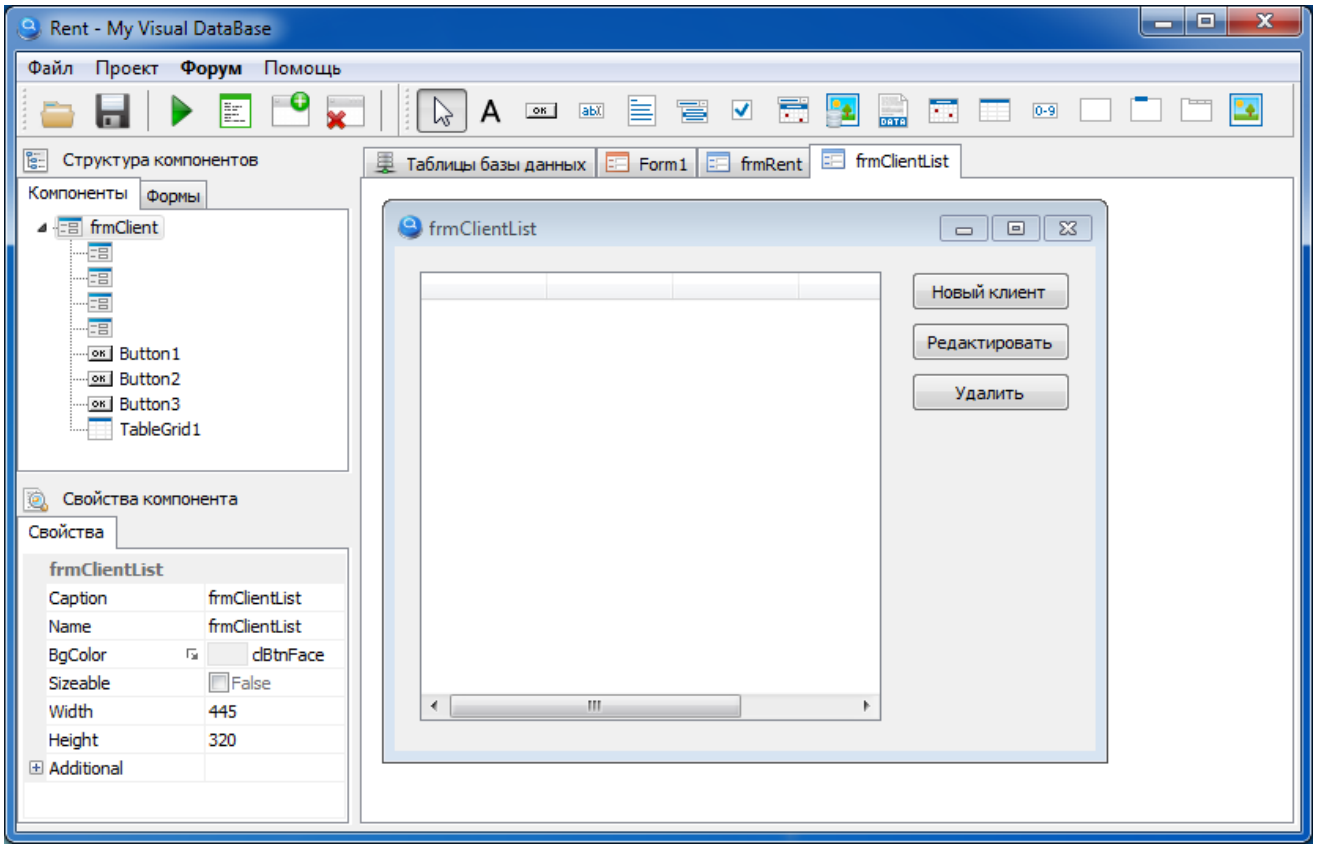


Рис.15

Создадим четвертую форму с именем **frmClient**, нажав кнопку  .

Данная форма предназначена для создания записи о новом клиенте, либо редактировании существующего.

Поместите компоненты на данную форму как показано на рисунке 16.

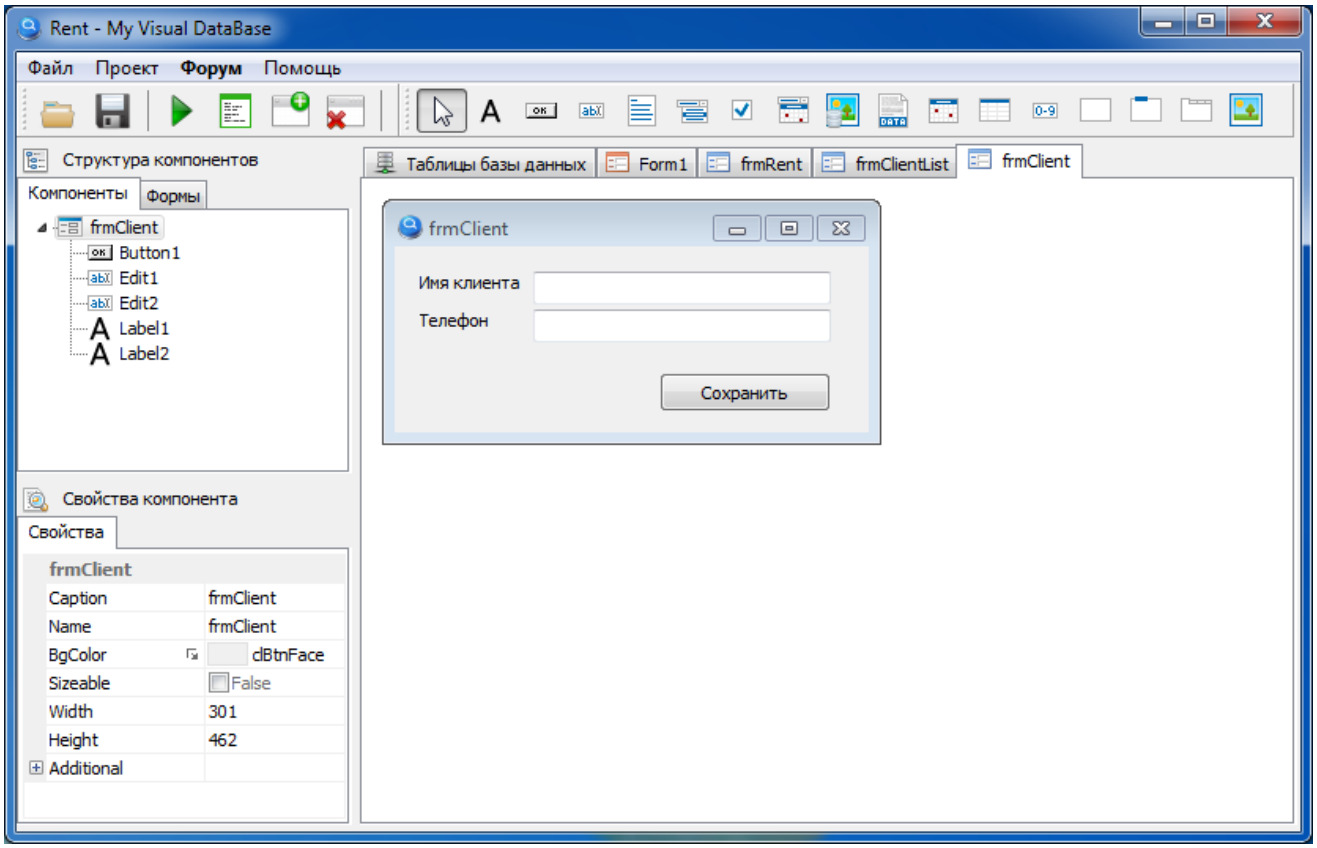


Рис.16

Создадим пятую форму с именем **frmTechList,** нажав кнопку  .

На создаваемой форме мы сможем видеть всю технику, которой вы располагаете, а так же вызвать форму для добавления новой техники, если вдруг такая появилась, либо редактировать информацию о существующей.

Поместите компоненты на данную форму как показано на рисунке 17.

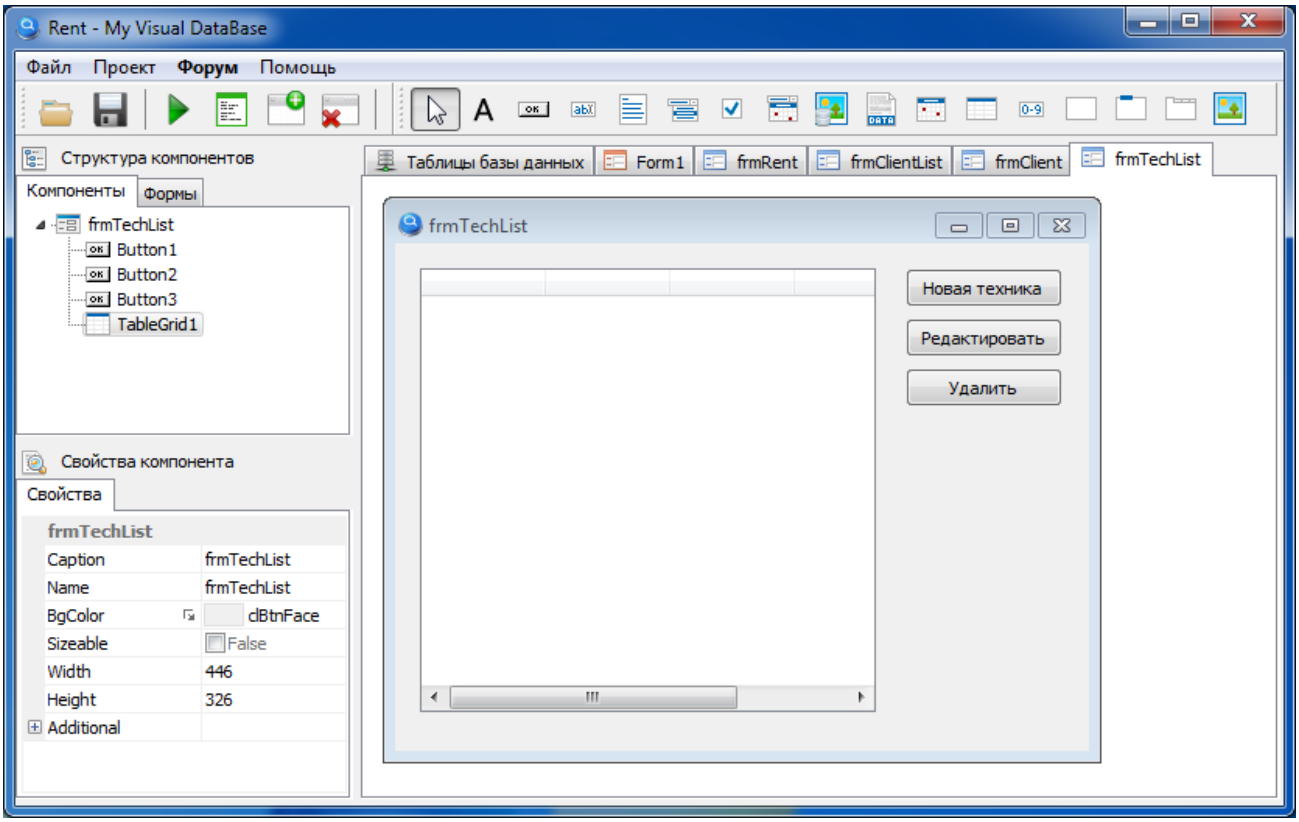


Рис.17

Создадим шестую, последнюю форму в нашем проекте, с именем **frmTech**, нажав кнопку  .

Данная форма предназначена для создания записи о новой технике, если такая появилась, либо редактирования существующей.

Поместите компоненты на данную форму как показано на рисунке 18.

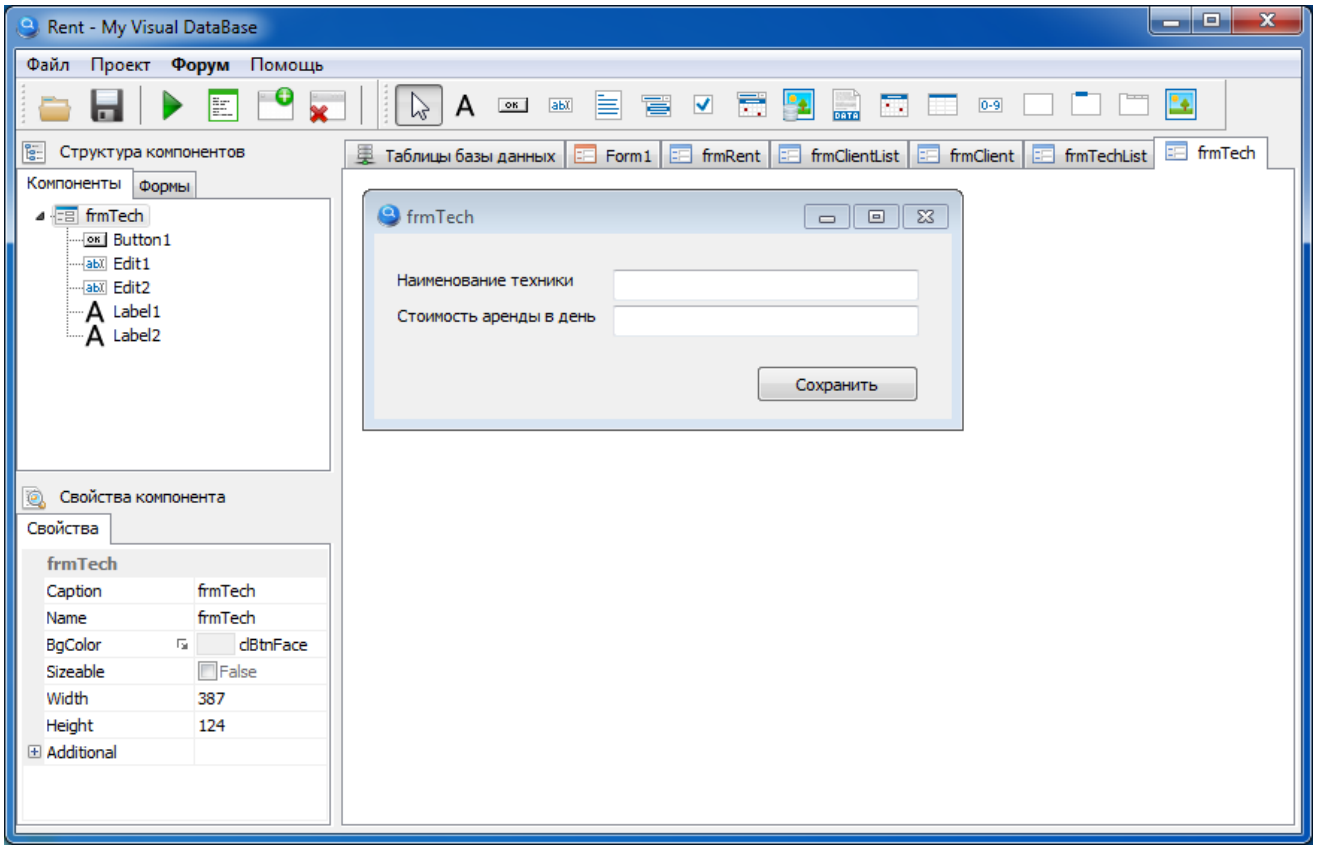


Рис.18

Если вы запустите проект, нажав на кнопку  , и попробуйте понажимать кнопки на форме, то заметите, что ничего не происходит. Причина в том, что компоненты на наших созданных формах пока не настроены, а для кнопок не присвоены действия. Поэтому переходим к последнему этапу создания приложения, к настройке компонентов на форме.

Как правило, настройка компонентов сводится к тому, чтобы кнопкам  присвоить действие, а компонентам предназначенные для ввода информации (такие как:  и т.д.) указать, к какой таблице БД и к какому полю они принадлежат.

Начнем с кнопок  . Задать действие для кнопки необходимо через панель «Свойства компонента», используя свойство **Action** (рис.19)

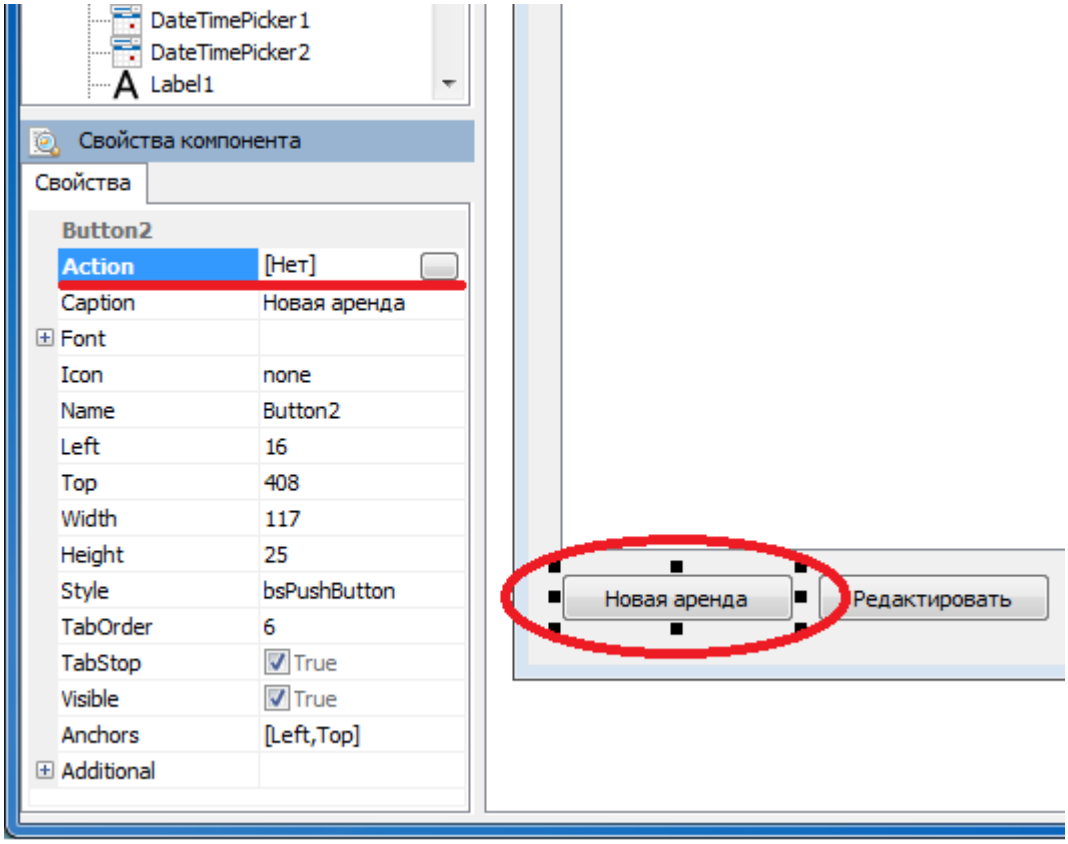


Рис.19

Перечислим доступные действия для кнопок, которые нам пригодятся для настройки нашего проекта:

- **Поиск** – позволяет найти записи в БД.
- **Новая запись** – вызывает форму для создания новой записи.
- **Сохранить запись** – сохраняет запись в БД и закрывает форму.
- **Показать запись** – вызывает форму и заполняет ее данными из БД.
- **Удалить запись** – удаляет запись, выбранную в компоненте таблицы.
- **Показать форму** – вызывает форму без дополнительных действий.
- **Закрыть форму** – закрывает форму.

Настройка формы «Form1»:

1. Кнопка **«Новая аренда»**

предназначена для создания новой записи, поэтому выбираем для данной кнопки действие **Новая запись**.

Данное действие вызовет форму, с помощью которой будет создана новая запись, для этого у нас есть форма с именем **frmRent.** Произведите настройку данной кнопки как показано на рисунке 20.

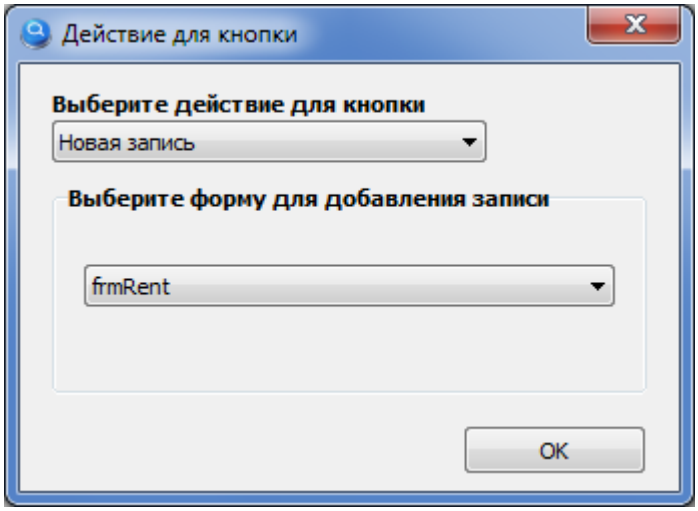
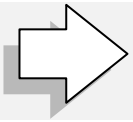


Рис.20



Порой пользователи вместо действия «Новая запись» выбирают действие «Показать форму», это ошибка, не перепутайте!

2. Кнопка **«Редактировать»**

предназначена для редактирования выбранной записи, которая будет выбрана пользователем в компоненте TableGrid

Выберите действие **«Показать запись»**. Для настройки данного действия, необходимо указать форму, которая будет использоваться для редактирования записи, и компонент TableGrid , в которой будет выбрана пользователем необходимая запись для редактирования. Настройте данную кнопку как показано на рисунке 21.

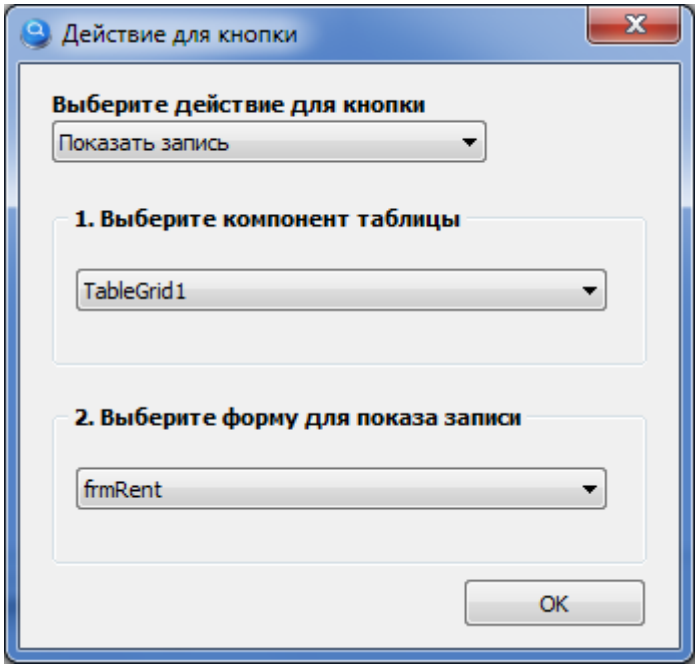


Рис.21

3. Кнопка **«Удалить»**

Предназначена для удаления выбранной записи, которая будет выбрана пользователем в компоненте TableGrid 

Настройте данную кнопку как показано на рисунке 22.

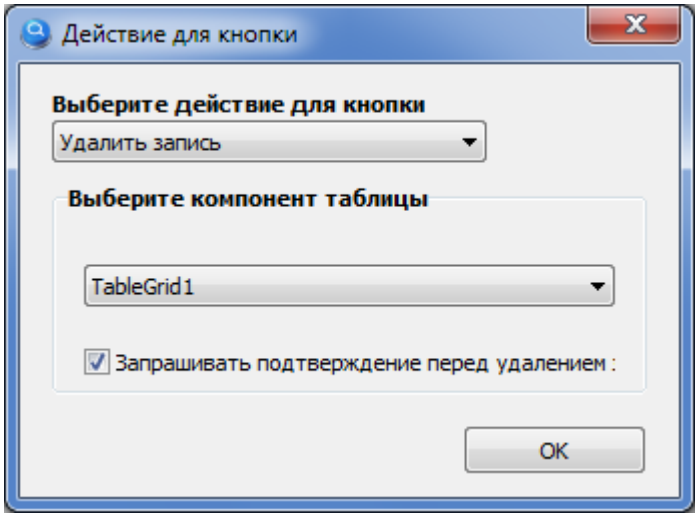


Рис.22

4. Кнопка **«Найти»**

Предназначена для поиска записей по произвольным критериям. Критерии для поиска вводятся в компоненты, такие как  . Каждый компонент может служить критерием для поиска в определенном поле БД, например, искать только по названию компании, либо по номеру телефона.

В зависимости от типа поля в БД, необходимо использовать соответствующий компонент. Ниже можете видеть, по каким типам полей в БД может быть использован компонент для поиска.

 - текстовое поле (TextBox). Тип поля БД: «ТЕКСТ» «ЦЕЛОЕ ЧИСЛО» «ВЕЩЕСТВ. ЧИСЛО» «ДЕНЬГИ»

 - выпадающий список (ComboBox). Поиск только по внешним ключам («СВЯЗЬ»)

 - флажок (CheckBox). Поиск только по полю с типом «ДА/НЕТ»

 - дата/время (DateTimePicker). Поиск по полям с типами «ДАТА/ВРЕМЯ» «ДАТА» «ВРЕМЯ»

 - таблица (TableGrid). Поиск только по внешним ключам («СВЯЗЬ»). Хотя компонент обычно используется для вывода результатов поиска, но может быть использован, как и критерий поиска по внешнему ключу, ведь по сути он похож на компонент  , т.к. тоже позволяет выбрать запись.

 - счетчик (Counter). Поиск только по полю с типом «Счетчик»

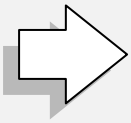
Теперь приступим к настройке данной кнопки, для этого выбираем действие **«Поиск»**. Настройка кнопки состоит из четырех шагов:

1. Выбрать компоненты, которые будут использоваться в качестве критерий для поиска записей. *В нашем случае выбираем следующие компоненты: ComboBox1 (Клиент), ComboBox2 (Техника), DateTimePicker1 (Начало аренды), DateTimePicker2 (Конец аренды).*

2. Выбрать таблицу БД, в которой будем искать. *Так как мы собираемся искать данные об аренде, выбираем таблицу БД **«Rent»***

3. Выбрать поля БД, которые необходимо видеть в качестве результата поиска в компоненте TableGrid  . *В нашем случае выберем следующие поля: Rent.DateStart, Rent.DateEnd, Rent.TotalCost, Client.Name, Equipment.Tech.*

4. Выбрать компонент таблицы (TableGrid), в который будет выведен результат поиска.



Чтобы кнопка с действием «Поиск», вывела все записи, достаточно нажать на нее с пустыми критериями поиска в компонентах.

На рисунке 23 можете видеть, как должна выглядеть настройка кнопки «Найти»

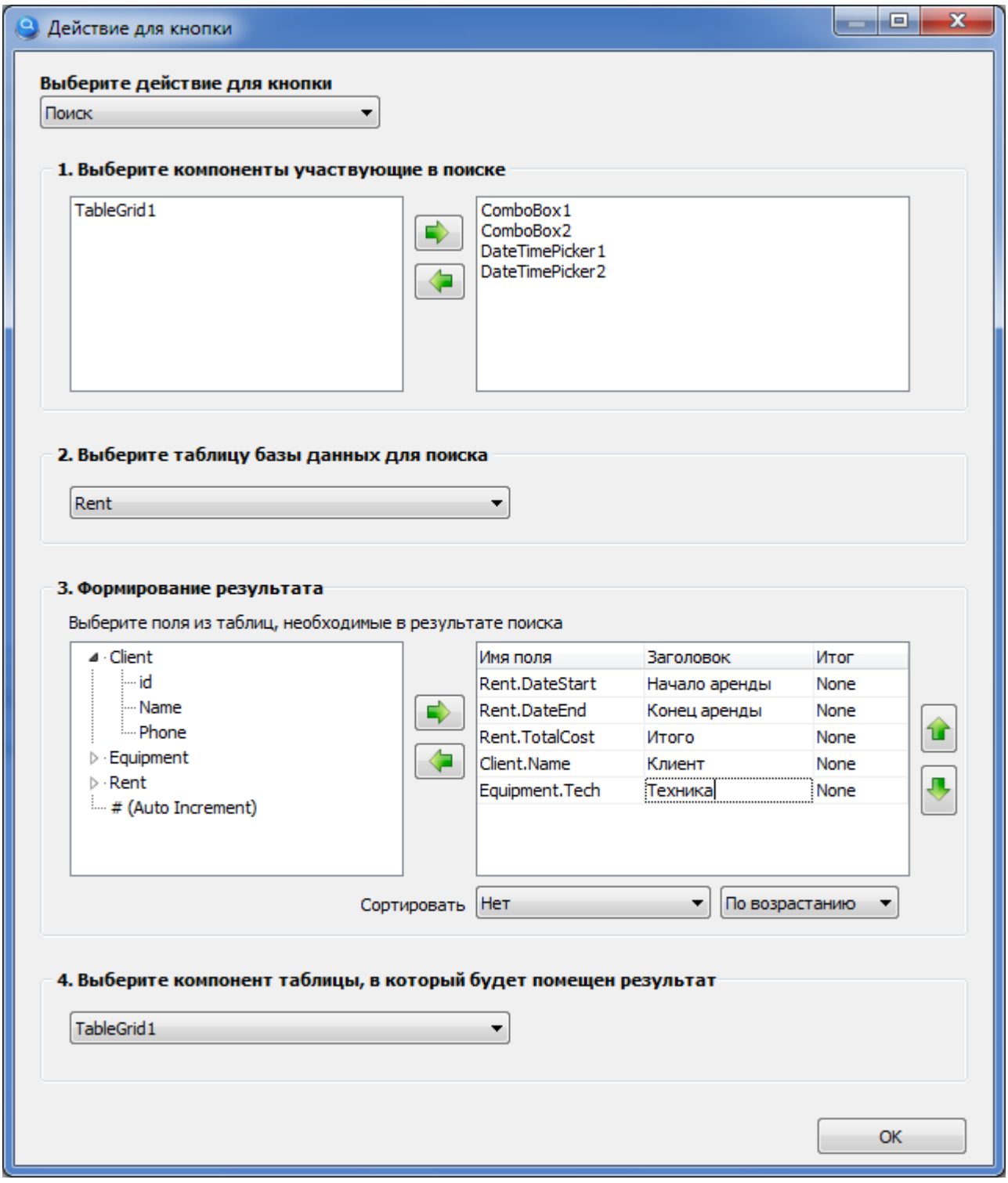


Рис.23

5. Кнопка **«Клиенты»**

Предназначена для показа формы, на которой мы сможем видеть всех наших клиентов. Выбираем действие **«Показать форму»**, а также выбираем форму **«frmClientList»**, которая будет показана при нажатии на данную кнопку (рис.24).

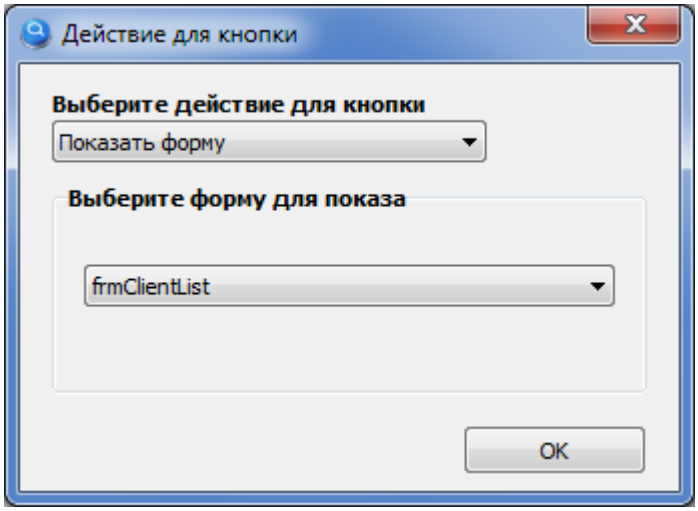


Рис.24

6. Кнопка **«Техника»**

Предназначена для показа формы, на которой мы сможем видеть всю технику, который мы располагаем. Выбираем действие **«Показать форму»**, а также выбираем форму **«frmTechList»**, которая будет показана при нажатии на данную кнопку (рис.25).

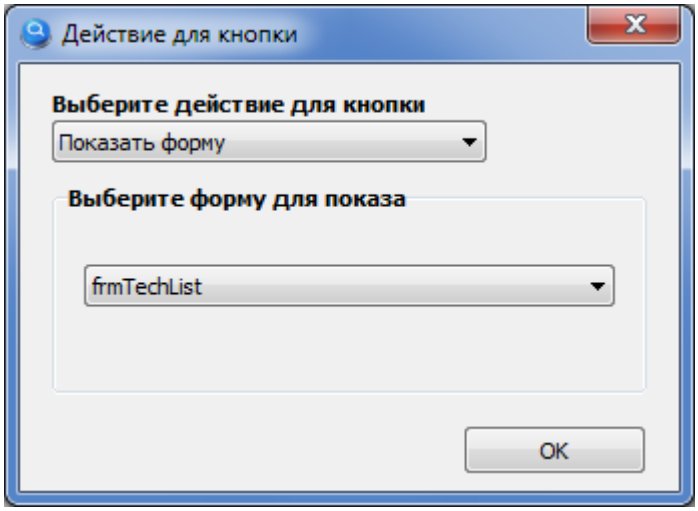


Рис.25

На этом настройка кнопок на форме завершена. Осталось настроить компоненты ComboBox  и DateTimePicker  , которые также присутствуют на форме **Form1**.

Настройка подобных компонентов, как правило, заключается лишь в указании, к какому конкретному полю в базе данных они принадлежат, для этого у компонентов присутствуют свойства **TableName** и **FieldName**, исключение составляет лишь компонент ComboBox  , вместо свойства **TableName** используется свойство **ForeignKey**, что переводится как внешний ключ. Таким образом, кнопка «Найти» будет знать, в каком поле БД необходимо искать, если пользователь введет критерий поиска в компонент. Приступим к настройке компонентов.

Начнем с компонента DateTimePicker1 (Начало аренды), информация о начале аренды содержится в таблице **Rent**, в поле **DateStart**, поэтому в свойстве **TableName** выбираем таблицу БД **Rent**, а в свойстве **FieldName** поле из этой таблицы **DateStart**, как показано на рисунке 26.

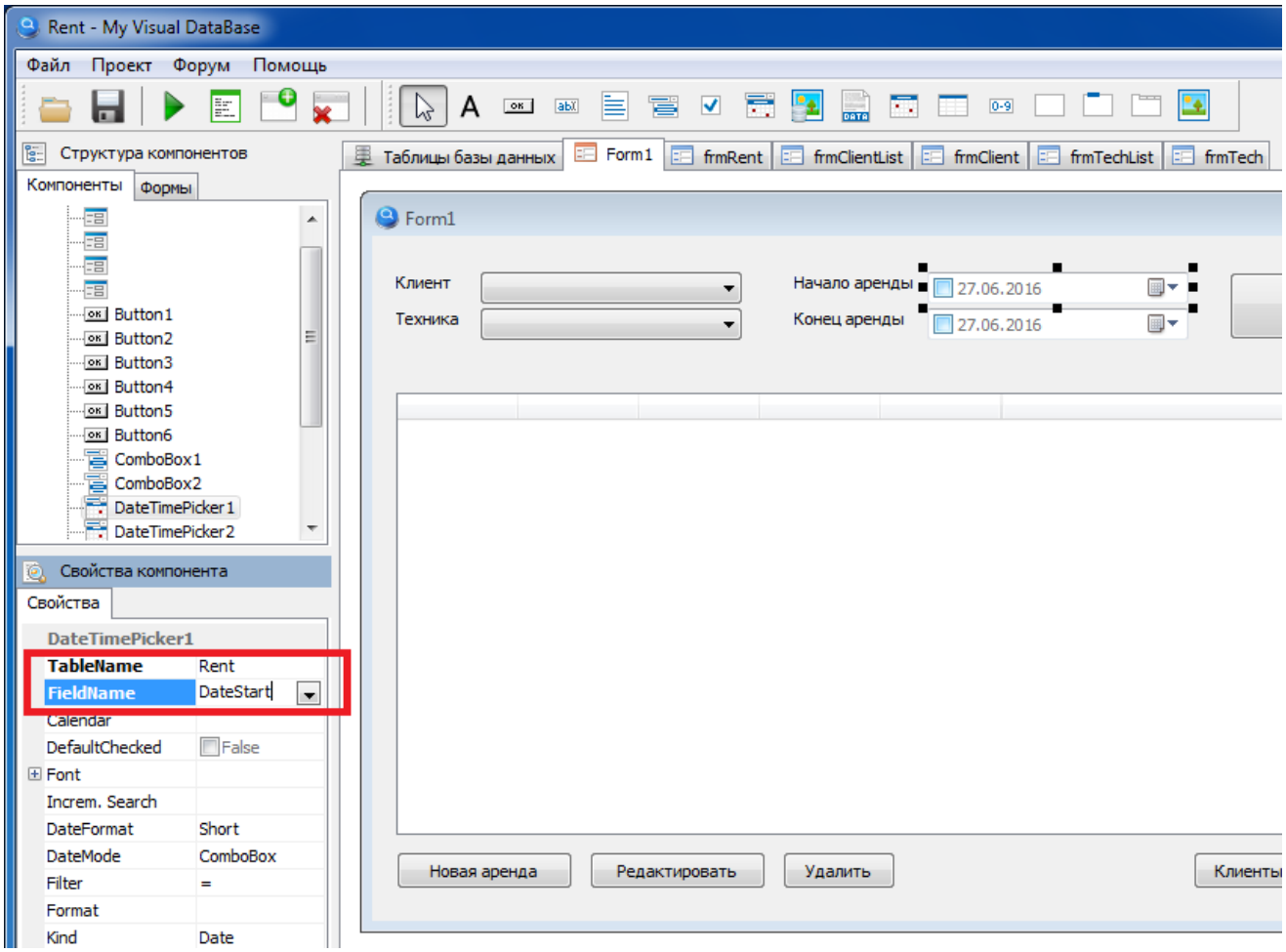


Рис.26

Для компонента DateTimePicker2 (Конец аренды) проделайте то же самое, но в свойстве **FieldName** выберите поле **DateEnd**.

Настройка компонентов ComboBox  , как уже было упомянуто выше, отличается тем, что вместо свойства **TableName** используется свойство **ForeignKey**. Данный компонент привязывается к внешнему ключу таблицы, таким образом, компонент показывает записи из таблицы, на которую ссылается внешний ключ, позволяя пользователю выбрать запись из данной таблицы.

Возможно, звучит запутано, поэтом давайте разберемся на примере, настроим компонент **ComboBox1** (Клиент).

Кнопка для поиска «Найти» настроена так, что производит поиск в таблице БД **Rent**, в данной таблице имеется внешний ключ **Rent.id_Client**, который нам и необходимо выбрать в свойстве **ForeignKey** компонента **ComboBox1** (Клиент). Внешний ключ **id_Client**, как видно из его названия ссылается на таблицу БД **Client**, поэтому в данном компоненте **ComboBox1** пользователь сможет выбрать из списка клиента. Останется выбрать поле БД в свойстве **FieldName**, в нашем случае это будет поле **Name**, таким образом, в компоненте будет виден список имен клиентов (рис. 27).

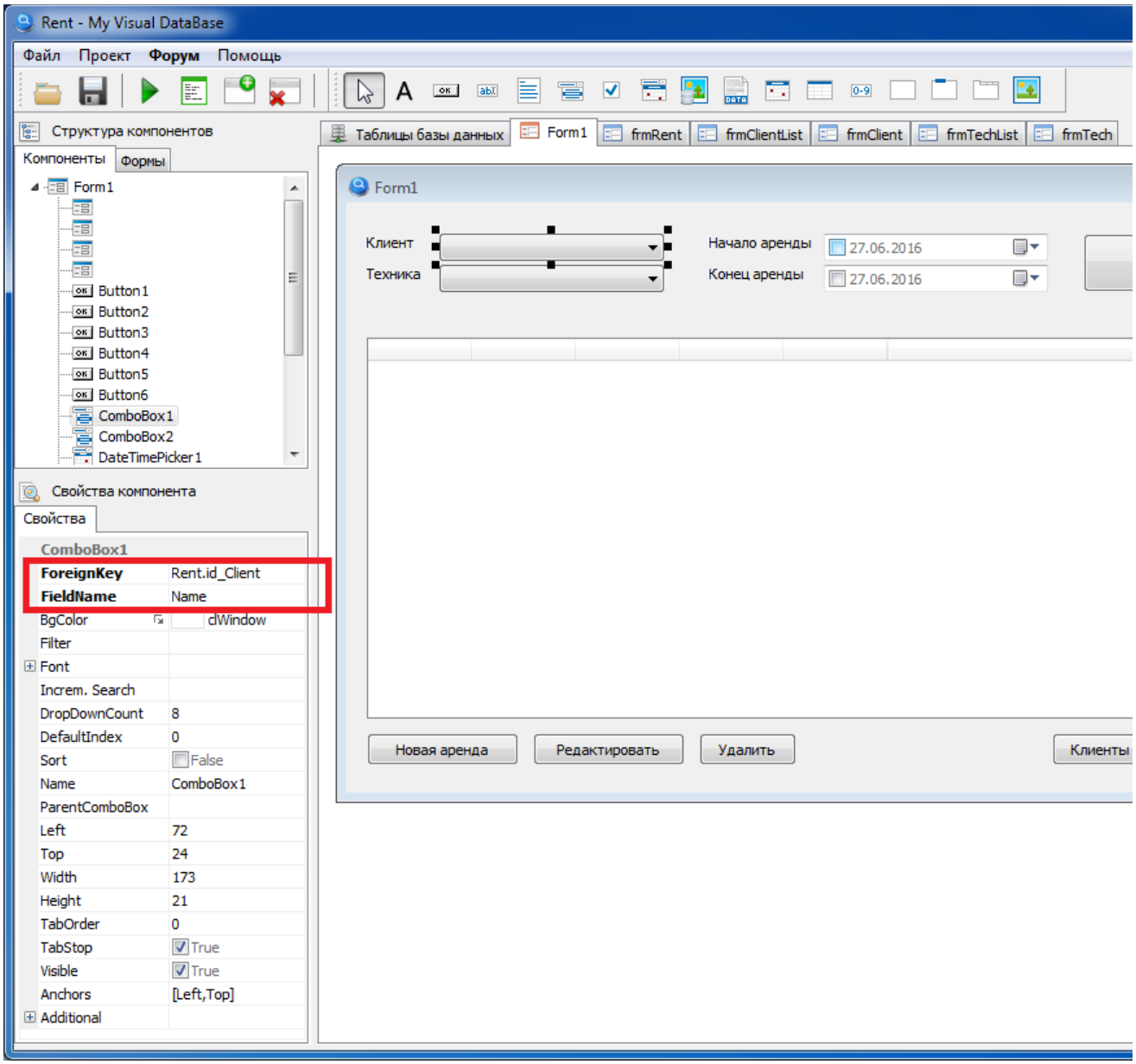


Рис.27

Таким же образом настраиваем компонент ComboBox2 (Техника), в свойстве **ForeignKey** выбираем внешний ключ **Rent.id_Equipment** и в свойстве **FieldName** выбираем поле БД **Tech**.

Пока мы настроили лишь одну форму, но не зависимо от формы и ее предназначения, настройка кнопок и компонентов довольно однообразна. Для кнопок необходимо выбрать правильное действие, а для компонентов выбрать, к какому полю БД они будут принадлежать, используя свойства

TableName и **FieldName**, в случае компонента **ComboBox**, вместо свойства **TableName** будет свойство **ForeignKey**.

Поэтому далее при настройке остальных форм, я лишь перечислю, какие действия кнопкам необходимо присвоить и какие поля БД присвоить компонентам на форме. Подробней остановлюсь лишь на настройке некоторых кнопок.

Настройка формы **frmRent**:

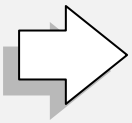
ComboBox1 (Клиент)
ForeignKey = Rent.id_Client
FieldName = Name

ComboBox2 (Техника)
ForeignKey = Rent.id_Equipment
FieldName = {Tech} {CostPerDay}руб. * подробней об этой записи ниже

DateTimePicker1 (Начало аренды)
TableName = Rent
FieldName = DateStart

DateTimePicker2 (Конец аренды)
TableName = Rent
FieldName = DateEnd

Edit1 (Стоимость)
TableName = Rent
FieldName = TotalCost



В свойстве FieldName компонента ComboBox вы можете указать сразу несколько полей БД, которые необходимо отобразить. Для этого их необходимо заключить в фигурные скобки.

Например, необходимо вывести наименование техники и ее стоимость за день:
{Tech} {CostPerDay}руб.

где Tech и CostPerDay названия полей в БД, остальной текст без фигурных скобок будет выведен без изменений. Таким образом, в ComboBox вы увидите: «Бульдозер ТМ-10 12000руб.»

Чуть подробней остановимся на кнопке **Button1** (Сохранить). Данная кнопка предназначена для сохранения данных в БД, поэтому для данной кнопки выбираем действие «Сохранить запись».

В настройках данной кнопки необходимо лишь выбрать, с каких компонентов на форме, будет сохранена информация в базу данных, в нашем случае необходимо выбрать все компоненты. Также необходимо выбрать таблицу БД, в которую будет сохранена информация, в нашем случае это таблица **Rent** (рис. 28).

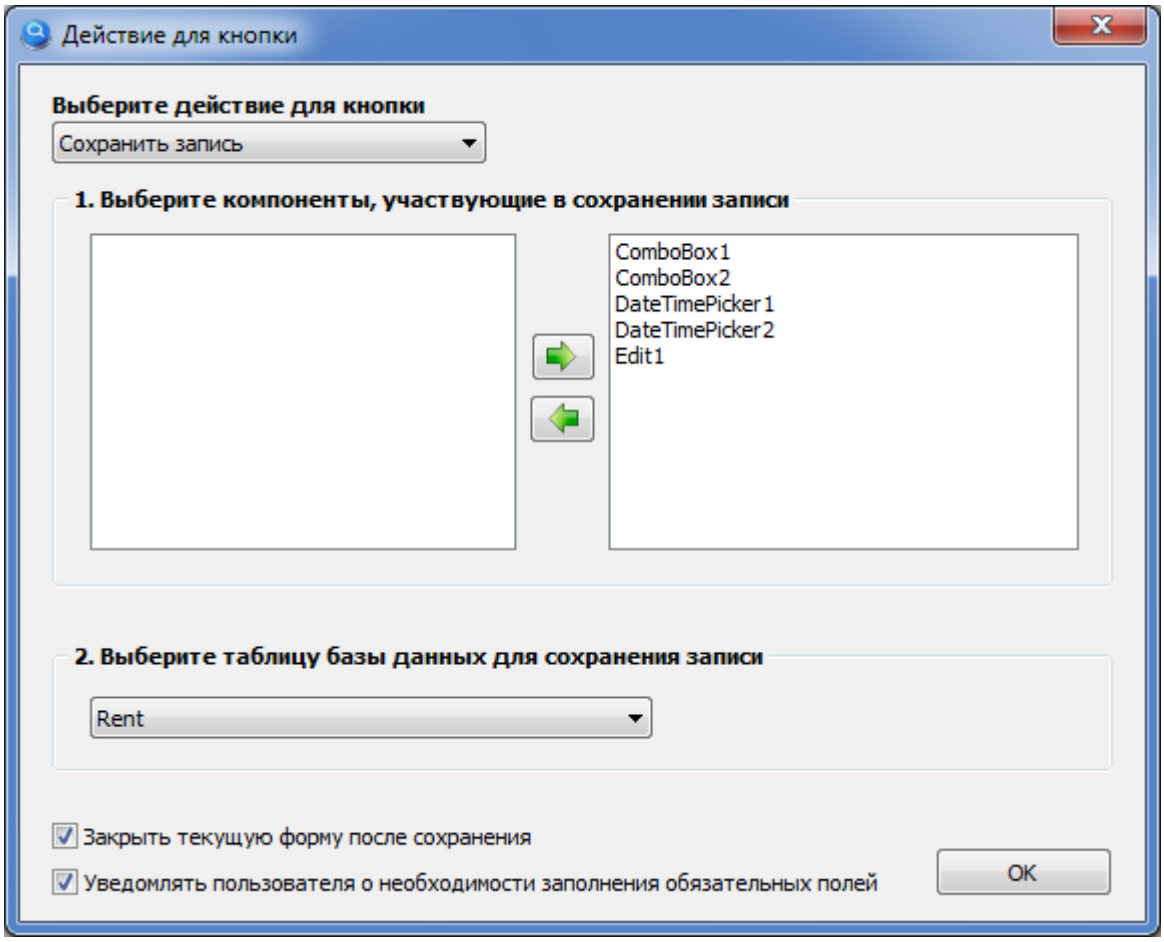


Рис.28

Также подробней остановимся на настройке кнопки Button2 (Закрыть).

Кнопка закрывает форму, без сохранения информации в базу данных. Поэтому просто выбираем действие «Закрыть форму». Какие либо настройки для данного действия не предусмотрены (рис. 29).

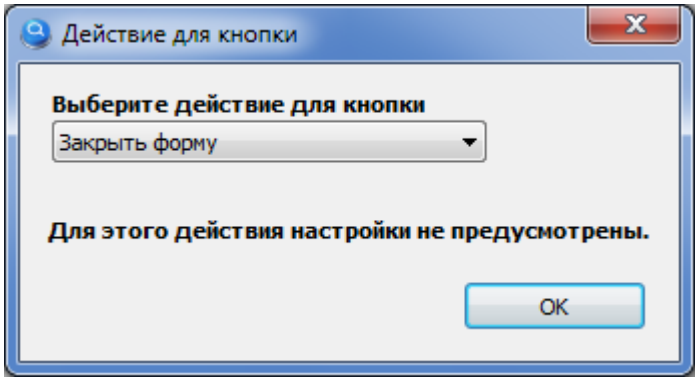


Рис.29

Настройка формы **frmClientList**:

Данная форма предназначена для просмотра всех наших клиентов, которые присутствуют в базе данных. Также на форме присутствуют кнопки для создания, редактирования или удаления клиента из БД.

Чтобы в компоненте таблицы появились все клиенты, которые присутствуют в БД, его необходимо настроить. Для вызова диалога настроек данного компонента, нажмите на панели «Свойства компонента» пункт «Настройка» (рис.30)

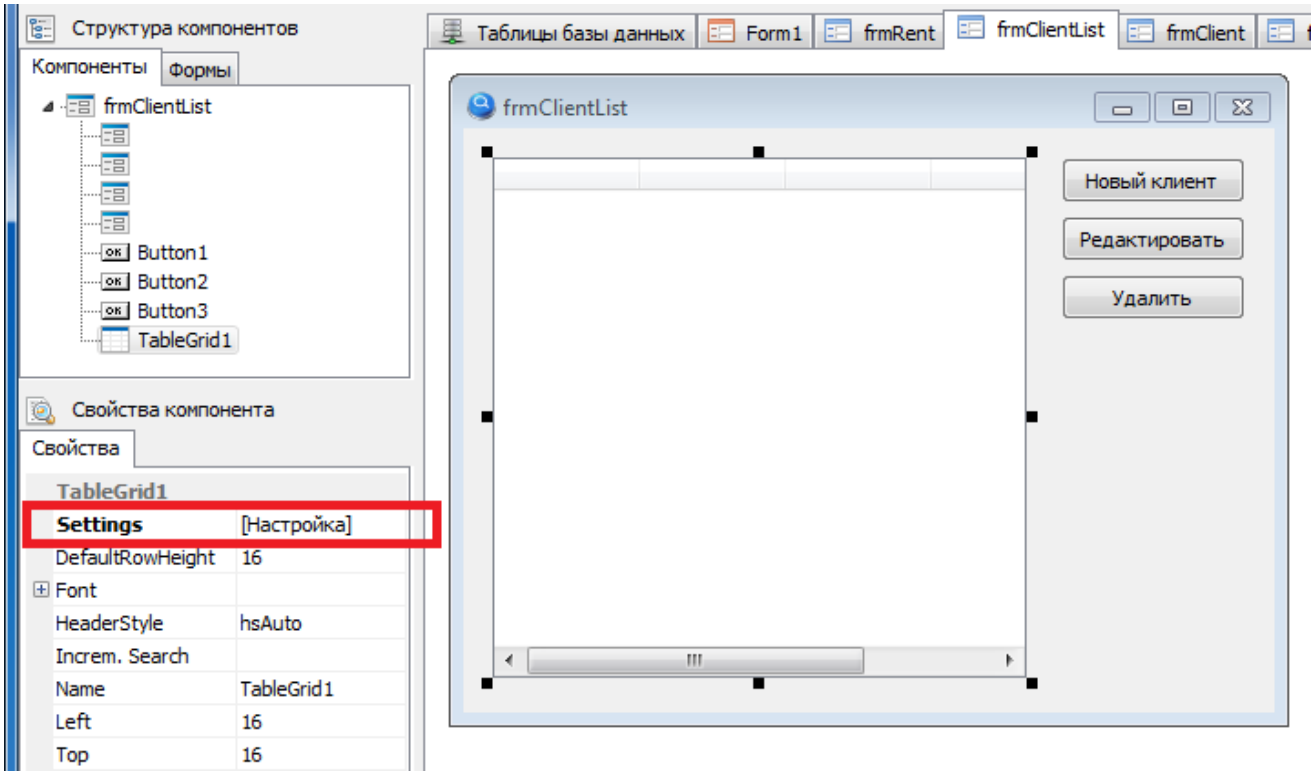


Рис.30

В настройках данного компонента необходимо выбрать таблицу БД, из которой необходимо вывести информацию и выбрать необходимые поля БД, которые мы хотели бы видеть в компоненте. Настройте данный компонент как показано на рисунке 31.

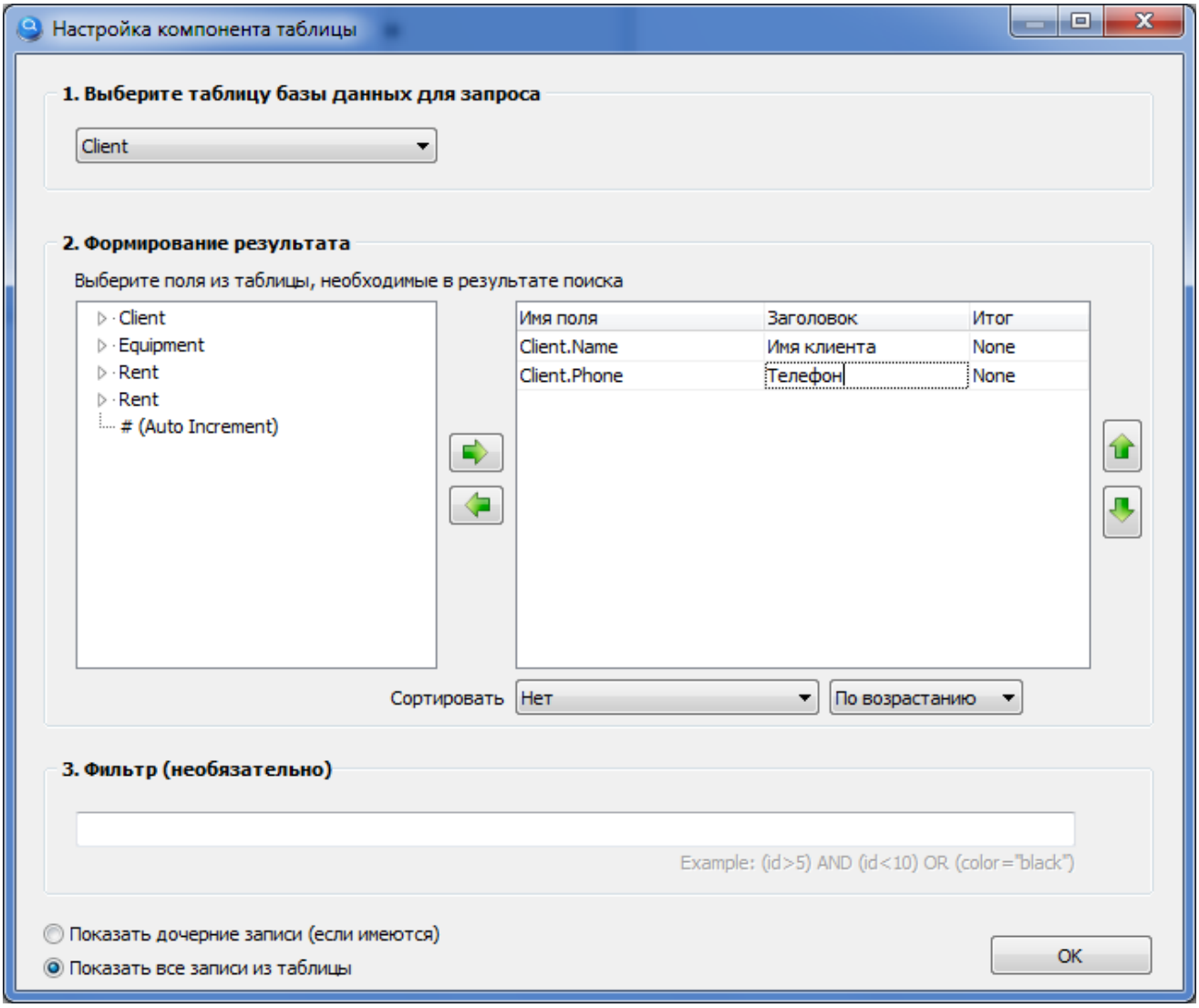


Рис.31

Обратите внимание на галочку с заголовком *«Показать все записи таблицы»*, она должна быть выбрана.

Далее просто перечислим настройки для кнопок создания, редактирования и удаления записи, т.к. мы уже подробно рассматривали подобные кнопки, при настройке формы Form1.

Новый клиент

Действие = Новая запись
Форма = frmClient

Редактировать

Действие = Показать запись
Компонент таблицы = TableGrid1
Форма = frmClient

Удалить

Действие = Удалить запись
Компонент таблицы = TableGrid1

Далее предлагаю вам самостоятельно настроить остальные формы, нужно лишь кнопкам присвоить правильные действия, а остальным компонентам верно указать свойства **TableName** и **FieldName**.

Форма **frmClient** предназначена для создания либо редактирования данных о клиенте, ее настройка аналогична настройке формы **frmRent.**

Настройка формы **frmTechList** аналогична настройке формы **frmClientList.**

Настройка формы **frmTech** аналогична настройке форм **frmRent** и **frmClient.**

После того, как вы закончили с настройками оставшихся форм, время запустить наш проект, нажав кнопку  и протестировать его работу.

Прежде чем вводить данные об аренде техники, необходимо в базу данных ввести всех ваших существующих клиентов, для этого нажмите кнопку «Клиенты», которая расположена на главной форме. Появится форма **frmClientList**, на которой вы можете создать клиентов, нажимая кнопку «Новый клиент» (рис. 32)

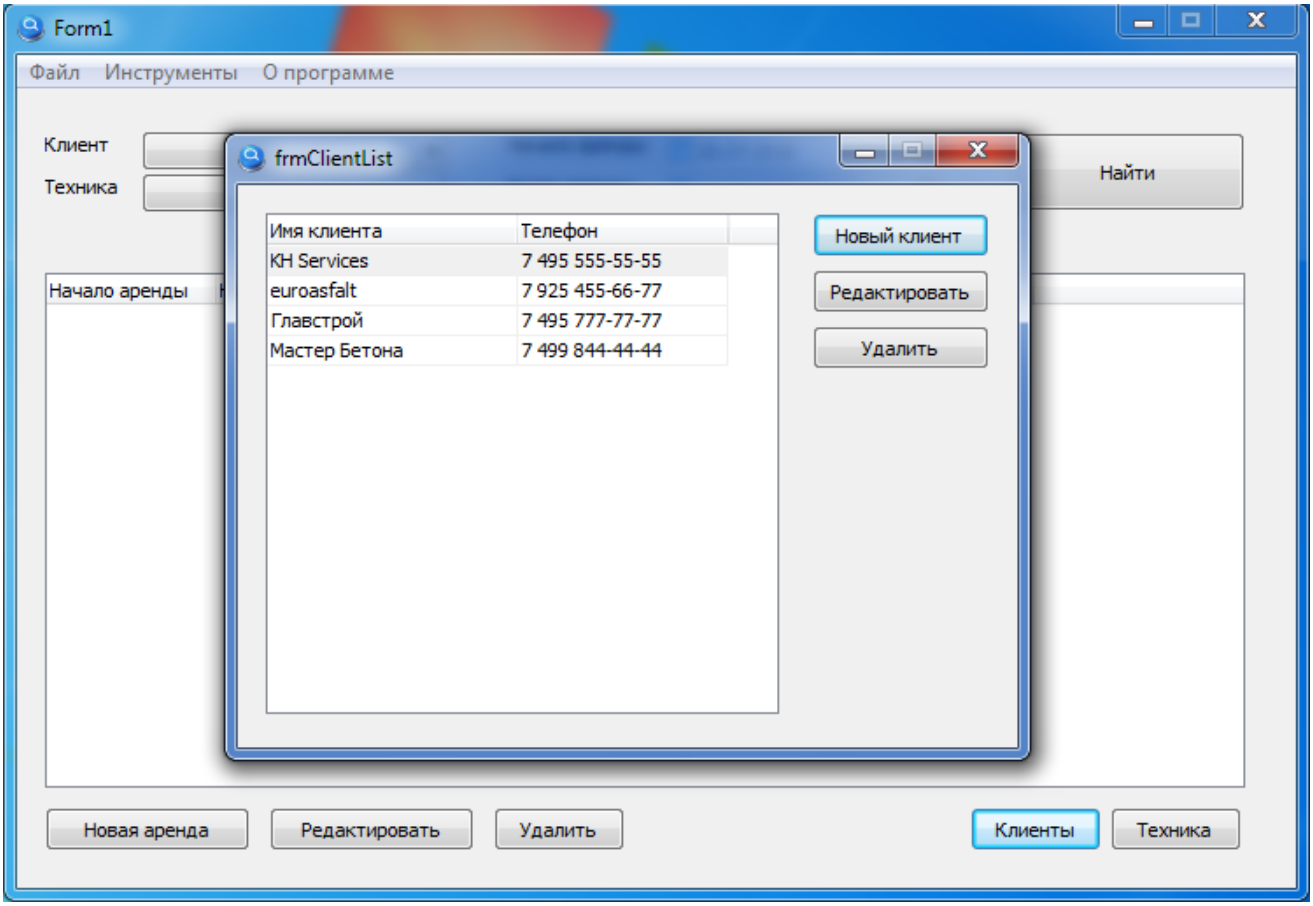


Рис.32

Также необходимо ввести в базу данных технику, которой вы располагаете, для этого на главной форме присутствует кнопка «Техника», которая вызовет форму **frmTechList**. Нажимая кнопку «Новая техника» введите необходимые данные (рис. 33).

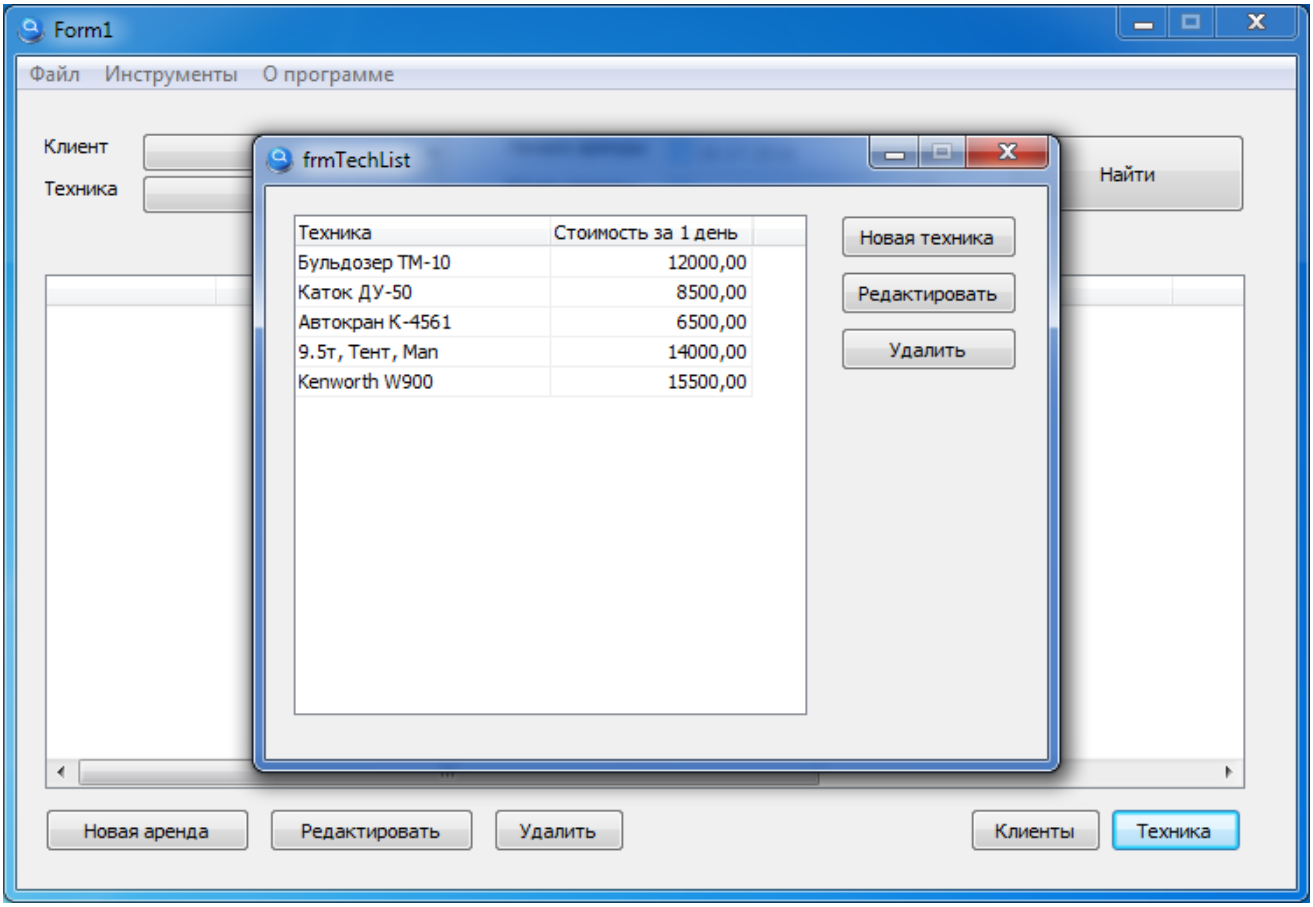


Рис.33

Приступим к вводу информации об аренде нашей техники. Для этого на главной форме нажмите «Новая аренда». Как видите теперь клиента и арендованную им технику мы просто выбираем из списка, в ручную остается ввести даты начала и конца аренды, а также итоговую стоимость аренды (рис. 34).

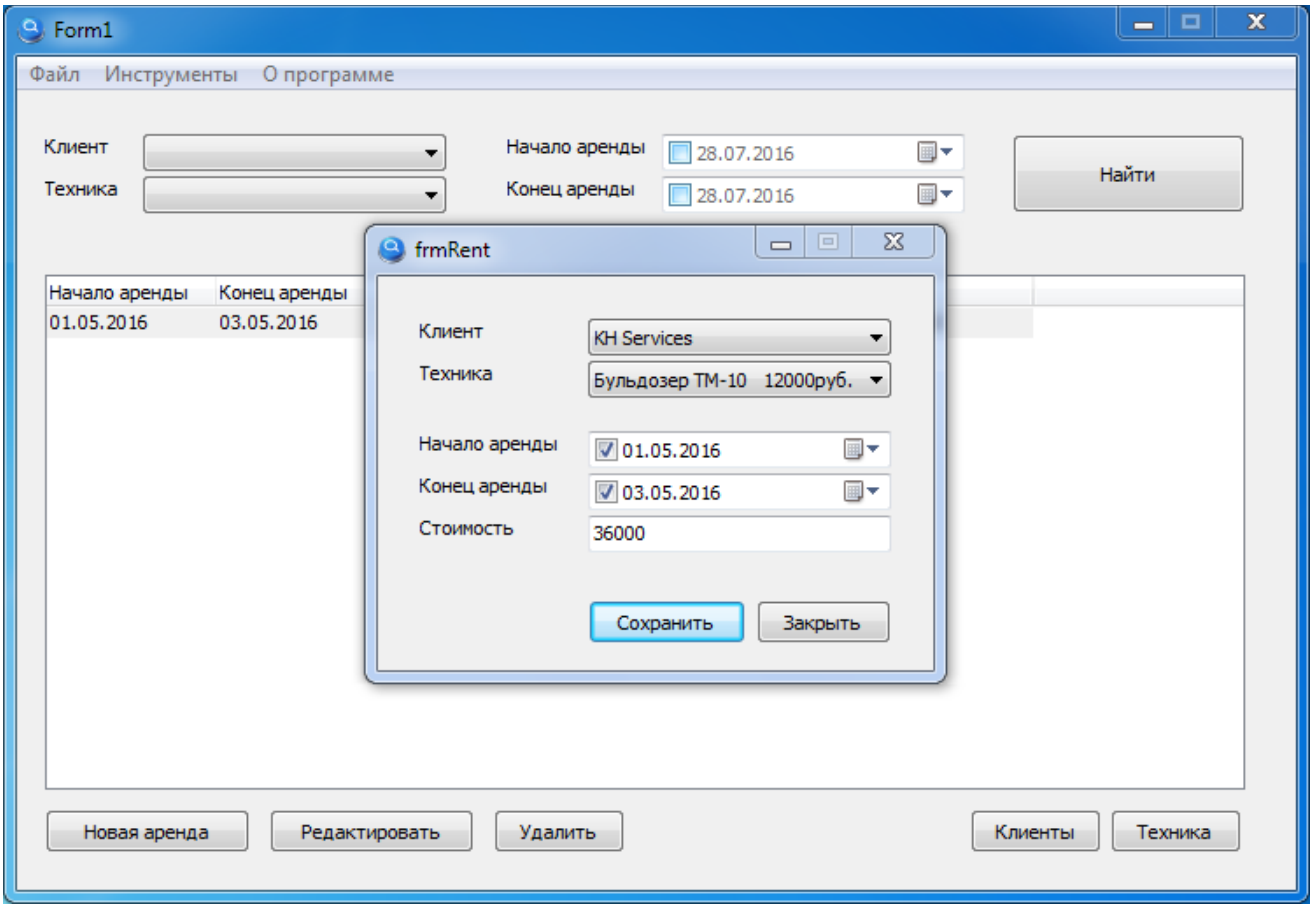
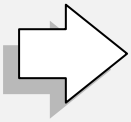


Рис.34



Возможно вы ожидали, что «Стоимость» посчитается автоматически, после того как укажите технику и сроки ее аренды, но чтобы это реализовать, необходимо использовать скрипты, работа с которыми будет рассмотрена чуть позже.

Пока нам придется «Стоимость» указать самостоятельно.

На рисунке 35 можете видеть главное окно программы с внесенными данными об аренде

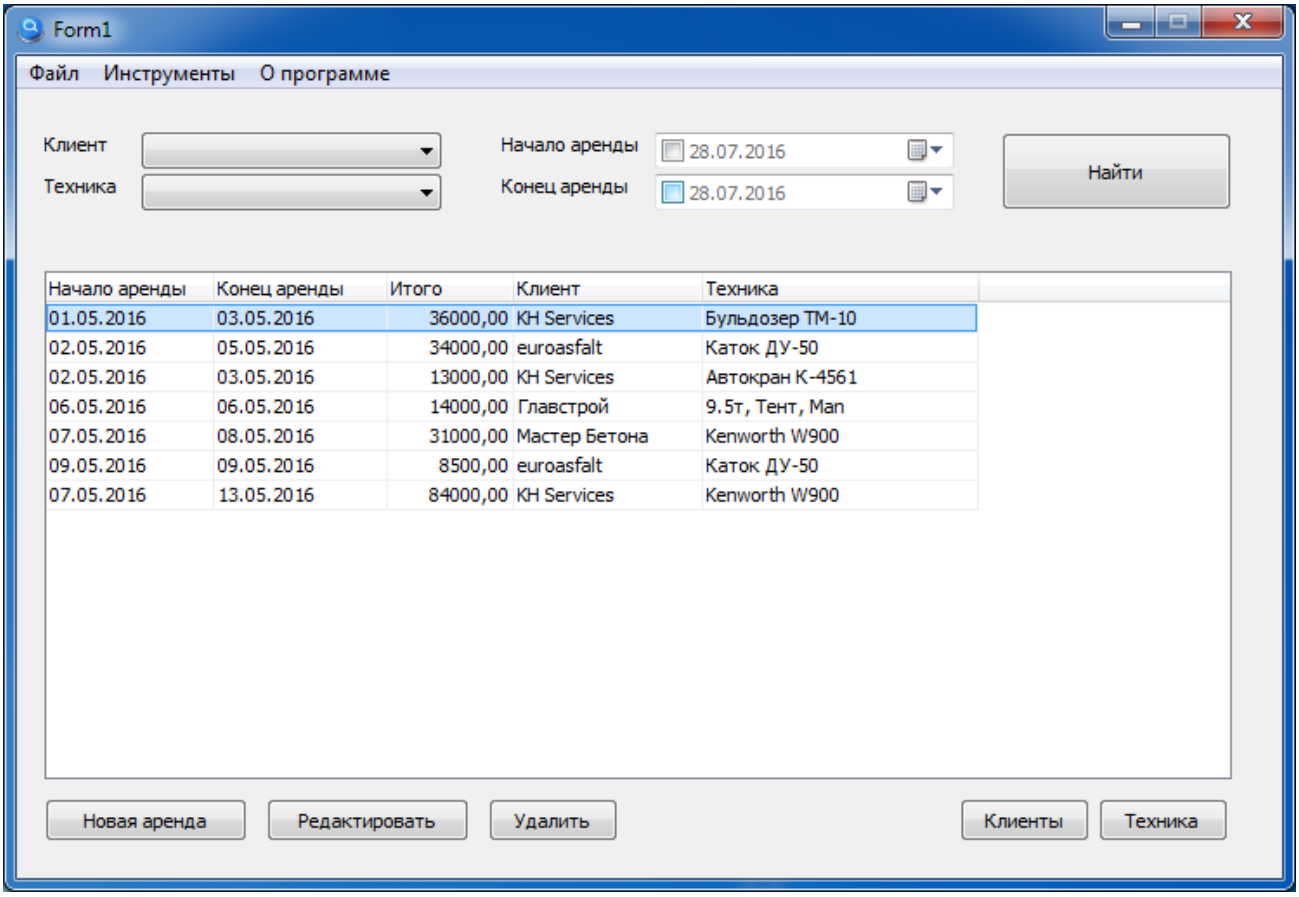


Рис.35

Теперь на главной форме вы можете воспользоваться поиском данных, например, выбрав клиента из выпадающего списка «Клиент», затем нажав кнопку «Найти», вы увидите все данные об аренде данного клиента.

Поздравляю! Ваше первое приложение для работы с базой данных готово!

Если вы откроете папку с проектом, то найдете в ней исполняемый файл, таким образом, ваш проект может работать на любом компьютере без необходимости установки программы My Visual Database. Достаточно просто скопировать папку вашего проекта со всеми его файлами.

Честно говоря, у созданного нами интерфейса есть недостаток.

Например, когда приходит новый клиент (информации о котором еще нет в БД) арендовать технику, нам придется сначала нажать кнопку «Клиенты», чтобы внести его данные. Затем на главной форме нажать кнопку «Новая аренда», чтобы уже на форме «frmRent» выбрать его из списка клиентов и только затем ввести данные об арендованной им технике. А теперь представьте, что клиент хочет арендовать 10 единиц различной техники? Т.е. нам придётся 10 раз выбирать одного и того же клиента из списка на форме «frmRent». Согласитесь, не удобно же!

Поэтому я предлагаю нам переделать данный проект, в котором мы создадим другой, более удобный интерфейс. Структура базы данных останется прежней.

Не подумайте, что это моя прихоть, на примере нового интерфейса мы рассмотрим возможность вывода дочерних записей в компонент TableGrid.

Итак, приступим.

Создайте новый проект (Меню: Файл > Новый проект) и сразу же сохраните его в отдельную папку, например «Rent project 2». Проекту дайте имя «Rent2», как показано на рисунке 36.

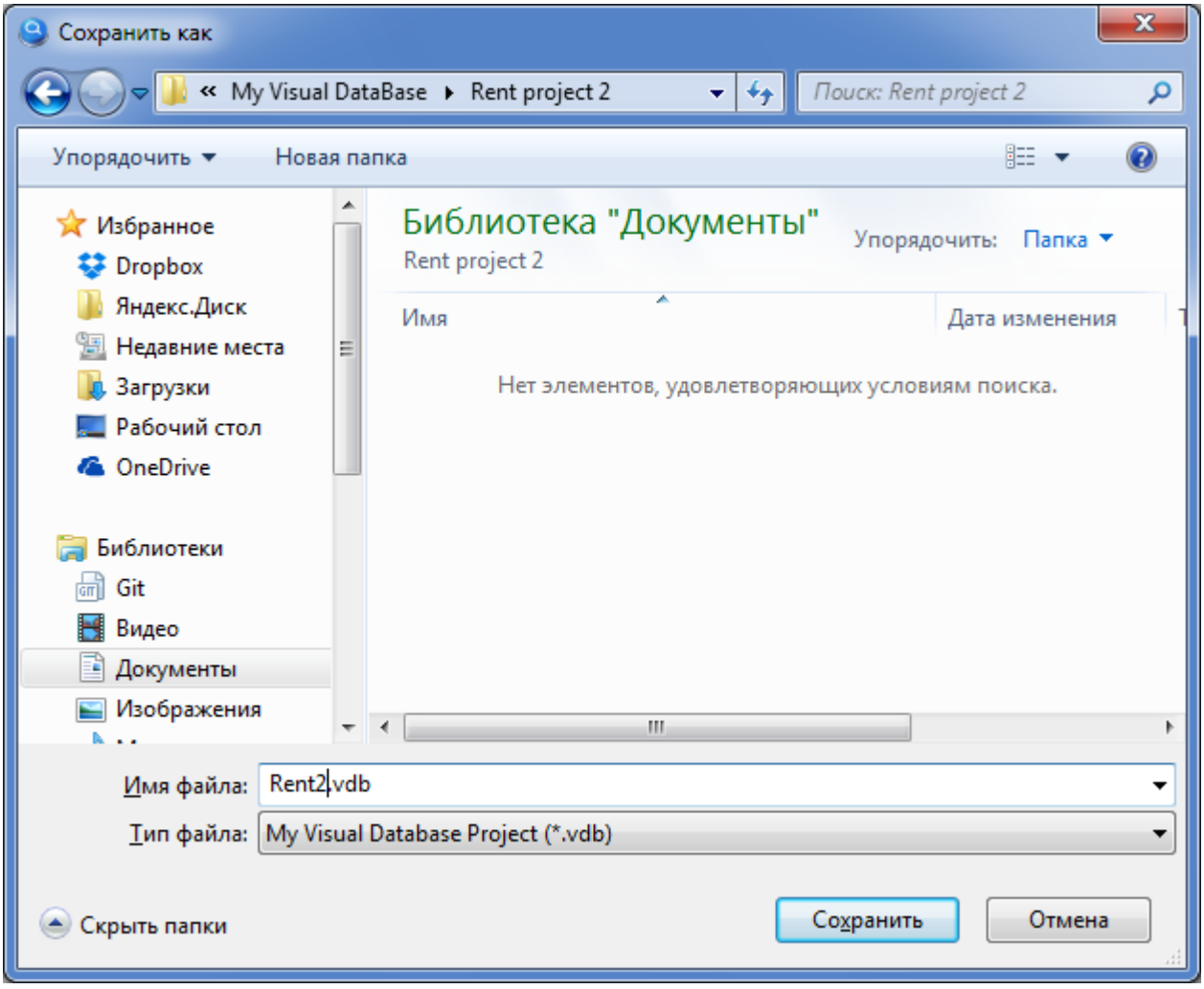


Рис.36

Далее нам следует создать точно такую же структуру базы данных на вкладке «Таблицы базы данных», как и в предыдущем нашем проекте, структуру БД можете видеть на рисунке 37.

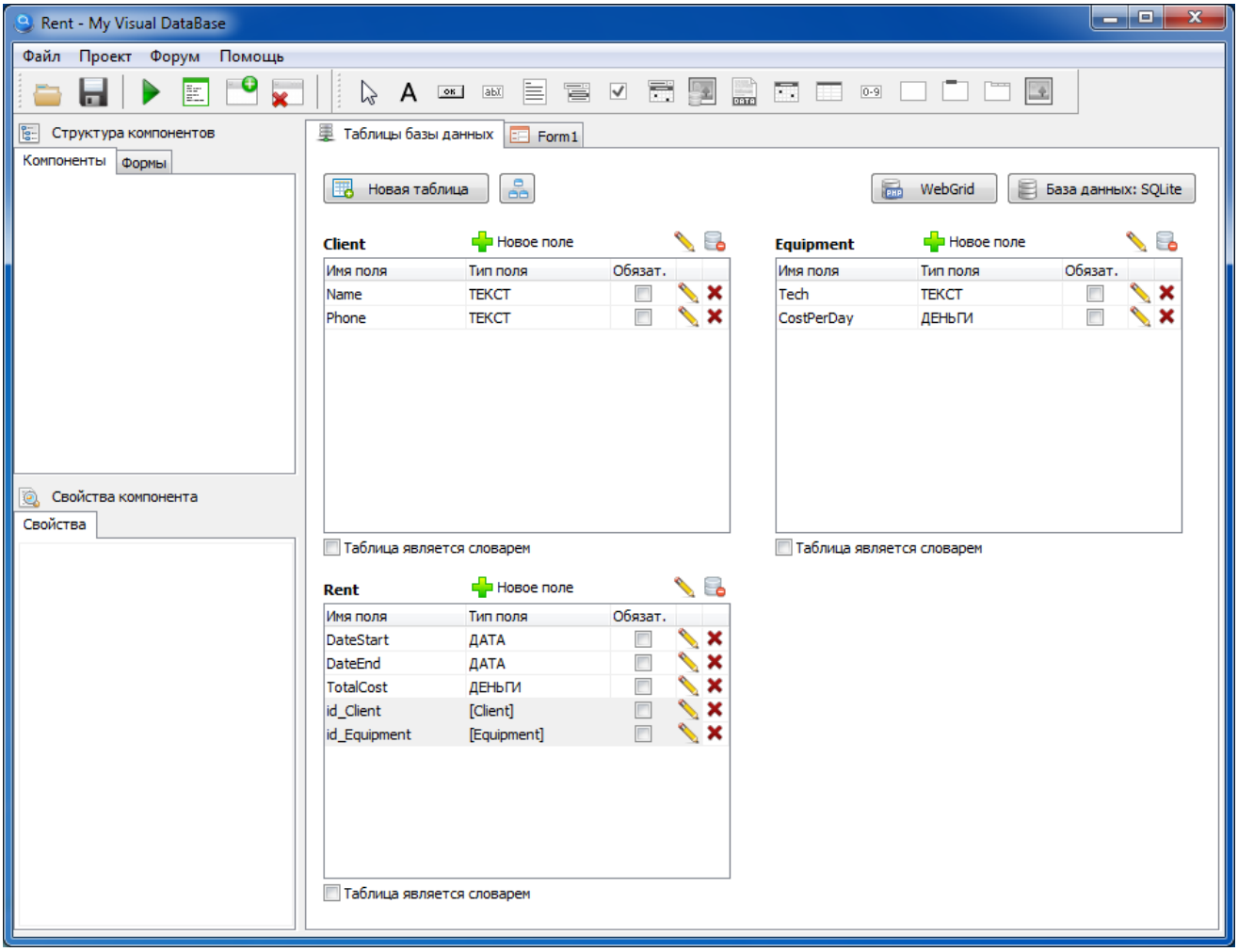


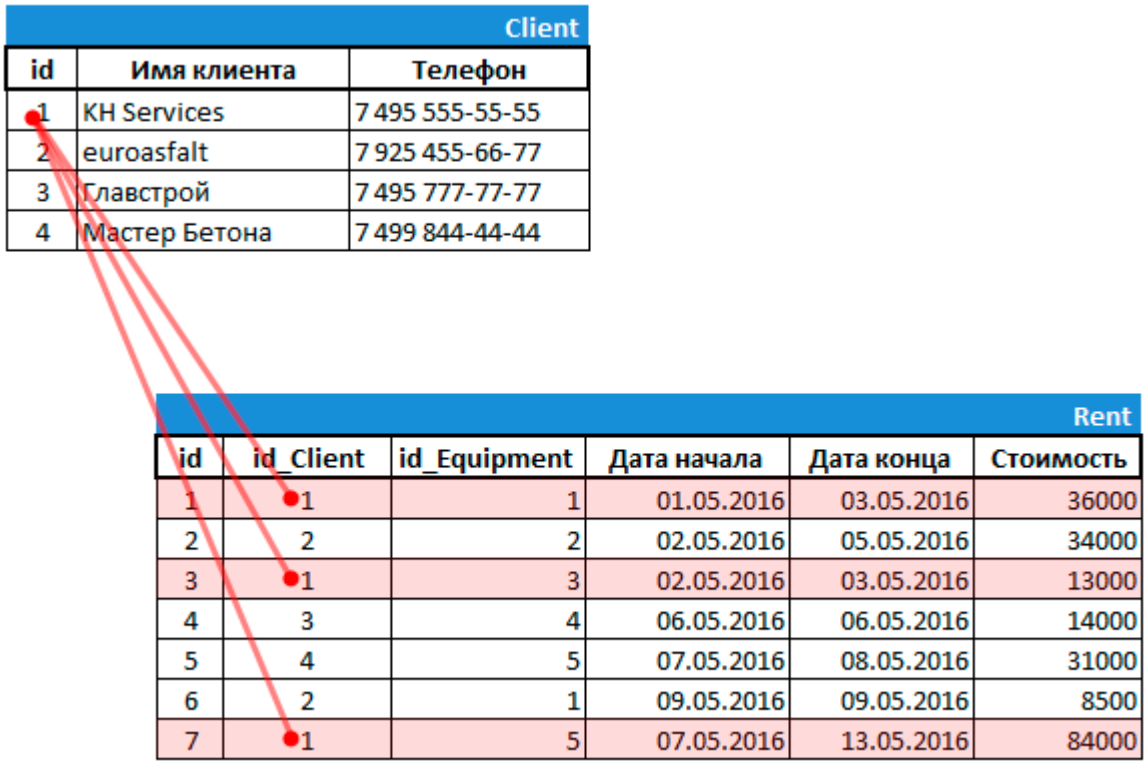
Рис.37

С процессом создания таблиц и полей мы уже знакомы, поэтому не будем подробно останавливаться на этом.

Переключитесь на вкладку «Form1» и поговорим о таком понятии, как дочерние записи.

Главным отличием нашего проекта, от его предыдущей версии будет в том, что на форме «frmClient», которая предназначена для создания/редактирования клиента, мы дополнительно разместим компонент TableGrid, который в свою очередь будет показывать записи об аренде техники, которую данный клиент арендовал. Другими словами в данном компоненте будут выведены **дочерние записи**, которые принадлежат клиенту.

Возможно, звучит немного запутано, поэтом взгляните на рисунок 38. Возьмём для примера клиента с именем «KH Services», который в свою очередь имеет записи об арендованной им техники в таблице «Rent», как видите, они отмечены красным, и именно эти записи являются **дочерними**, по отношению к клиенту «KH Services» из таблицы «Rent». Также обратите внимание, что у дочерней записи всегда есть внешний ключ, который ссылается на родительскую запись.



Client

id	Имя клиента	Телефон
1	KH Services	7 495 555-55-55
2	euroasfalt	7 925 455-66-77
3	Главстрой	7 495 777-77-77
4	Мастер Бетона	7 499 844-44-44

Rent

id	id_Client	id_Equipment	Дата начала	Дата конца	Стоимость
1	1	1	01.05.2016	03.05.2016	36000
2	2	2	02.05.2016	05.05.2016	34000
3	1	3	02.05.2016	03.05.2016	13000
4	3	4	06.05.2016	06.05.2016	14000
5	4	5	07.05.2016	08.05.2016	31000
6	2	1	09.05.2016	09.05.2016	8500
7	1	5	07.05.2016	13.05.2016	84000

Рис.38

И чтобы стало окончательно понятно, забегу немного вперед и покажу, как будет выглядеть форма «frmClient» уже в готовом проекте (рис. 39)

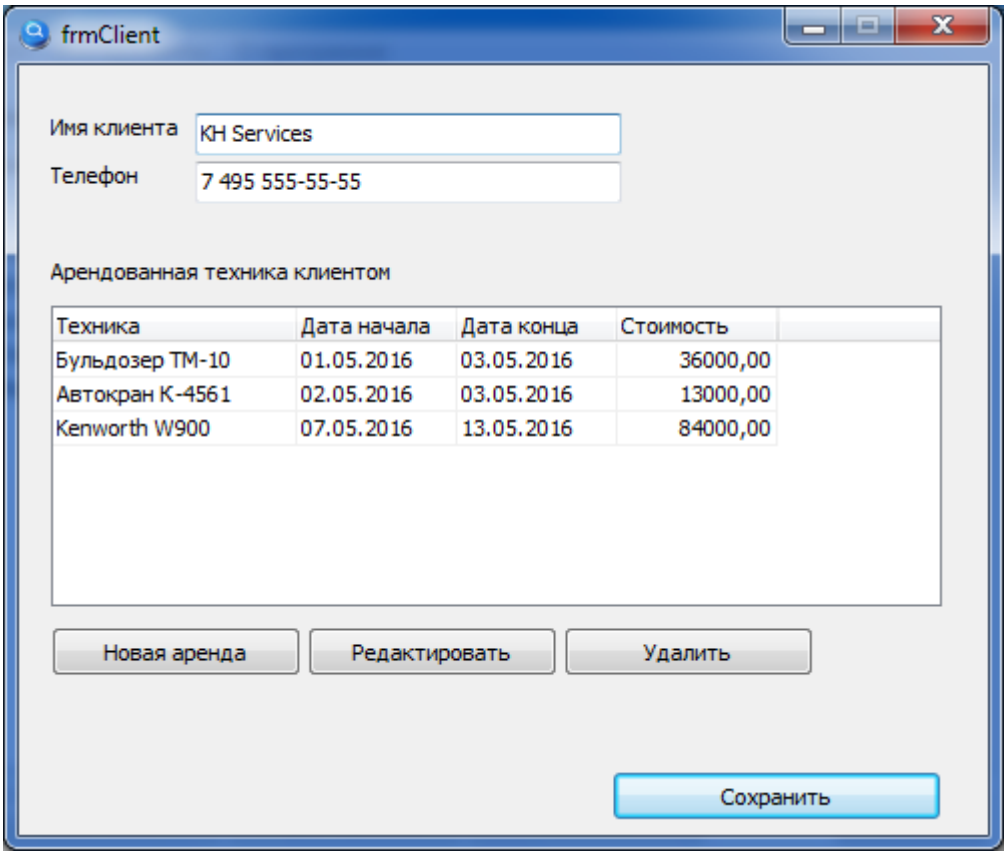


Рис.39

Чтобы сделать использование программы более удобным, предлагаю на главной форме («Form1») показывать список клиентов, вместо записей об аренде. Также на форме для создания/редактирования клиента («frmClient») разместить компонент TableGrid, который будет показывать арендованную технику клиента за все время. Если выражаться формальным языком, показывать дочерние записи клиента. Также на данной форме будут кнопки для добавления/редактирования/удаления арендованной техники данным клиентом.

Таким образом, когда к нам приходит новый клиент, нам достаточно будет на форме «Form1» нажать кнопку «Новый клиент», после чего появится форма «frmClient», где мы введем его данные и на этой же форме нажмем кнопку «Новая аренда», что вызовет появление формы «frmRent», в которой мы выбираем арендованную технику, даты аренды и ее стоимость.

Также если клиенту необходимо арендовать 10 единиц техники, нам не придется 10 раз выбирать данного клиента, чтобы создать запись об аренде.

Приступим к создания форм, давайте перечислим их.

1. **Form1**
   Форма для поиска клиентов. Эта же форма будет главной, именно она будет появляться при запуске вашего приложения.

2. **frmClient**
   Форма создания и редактирования клиента. Данная форма будет записывать данные в таблицу БД Client. Также на форме будет виден список арендованной техники данным клиентом.

3. **frmRent**
   Форма для создания и редактирования записи об аренде. Данная форма будет записывать данные в таблицу БД Rent.

4. **frmTechList**
   Форма со списком строительной техники.

5. **frmTech**
   Форма создания и редактирования информации о строительной техники. Данная форма будет записывать данные в таблицу БД Equipment.

Как видите, теперь нам нужно на одну форму меньше, что уже неплохо.

Приступим…

Форма **Form1**:

Расположите компоненты на форме Form1 как показано на рисунке 40.

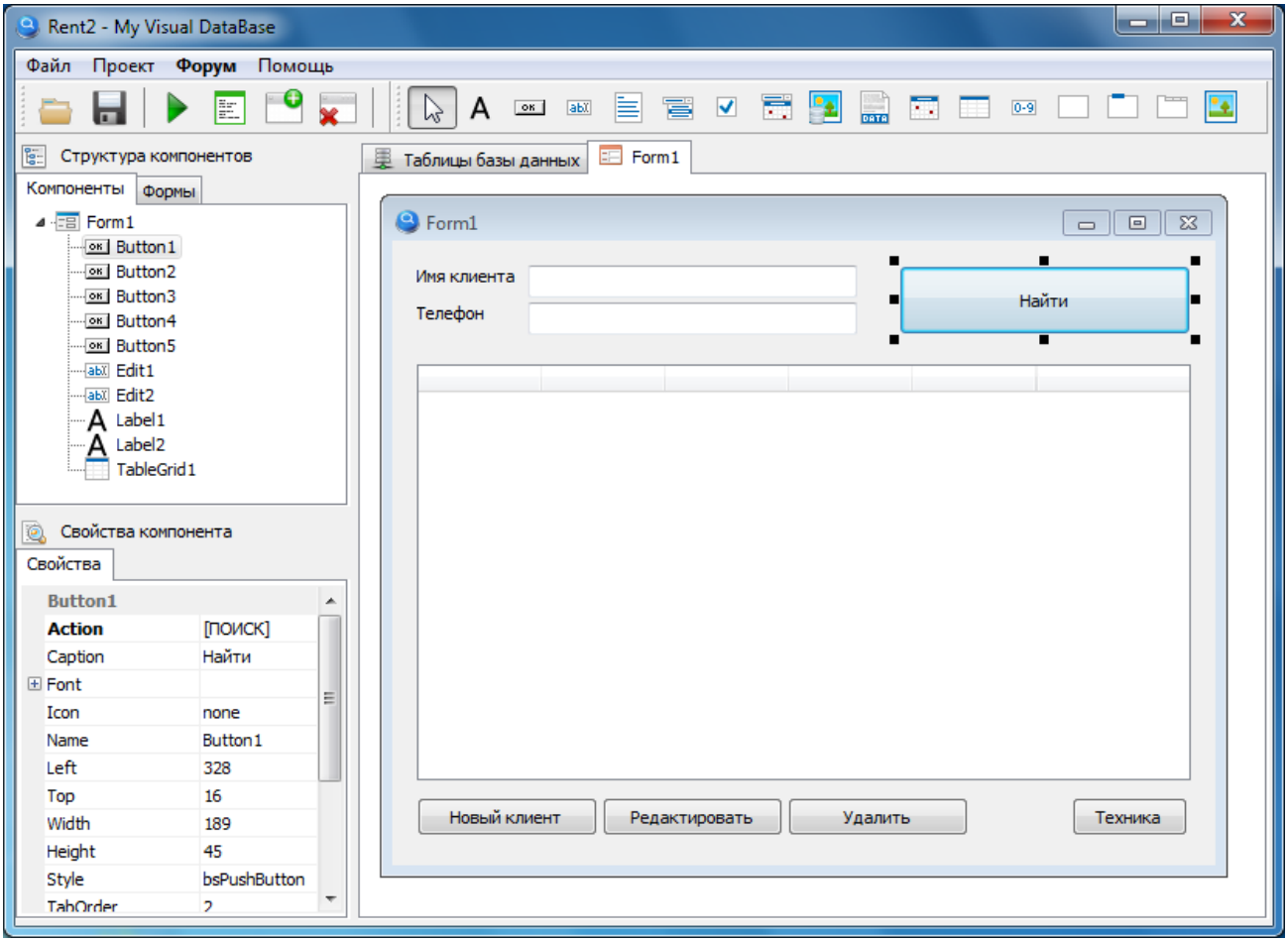


Рис.40

На данной форме вы сможете найти клиента по его имени либо телефонному номеру. Также на форме присутствуют кнопки, для вызова формы для создания либо редактирования клиента, также удаление клиента из базы данных. И кнопка с именем «Техника», которая вызовет другую форму, со списком техники, которой вы располагаете.

Форма **frmClient**:

Создайте новую форму, нажав кнопку  . Введите название формы **frmClient**

Расположите компоненты на данной форме как показано на рисунке 41.

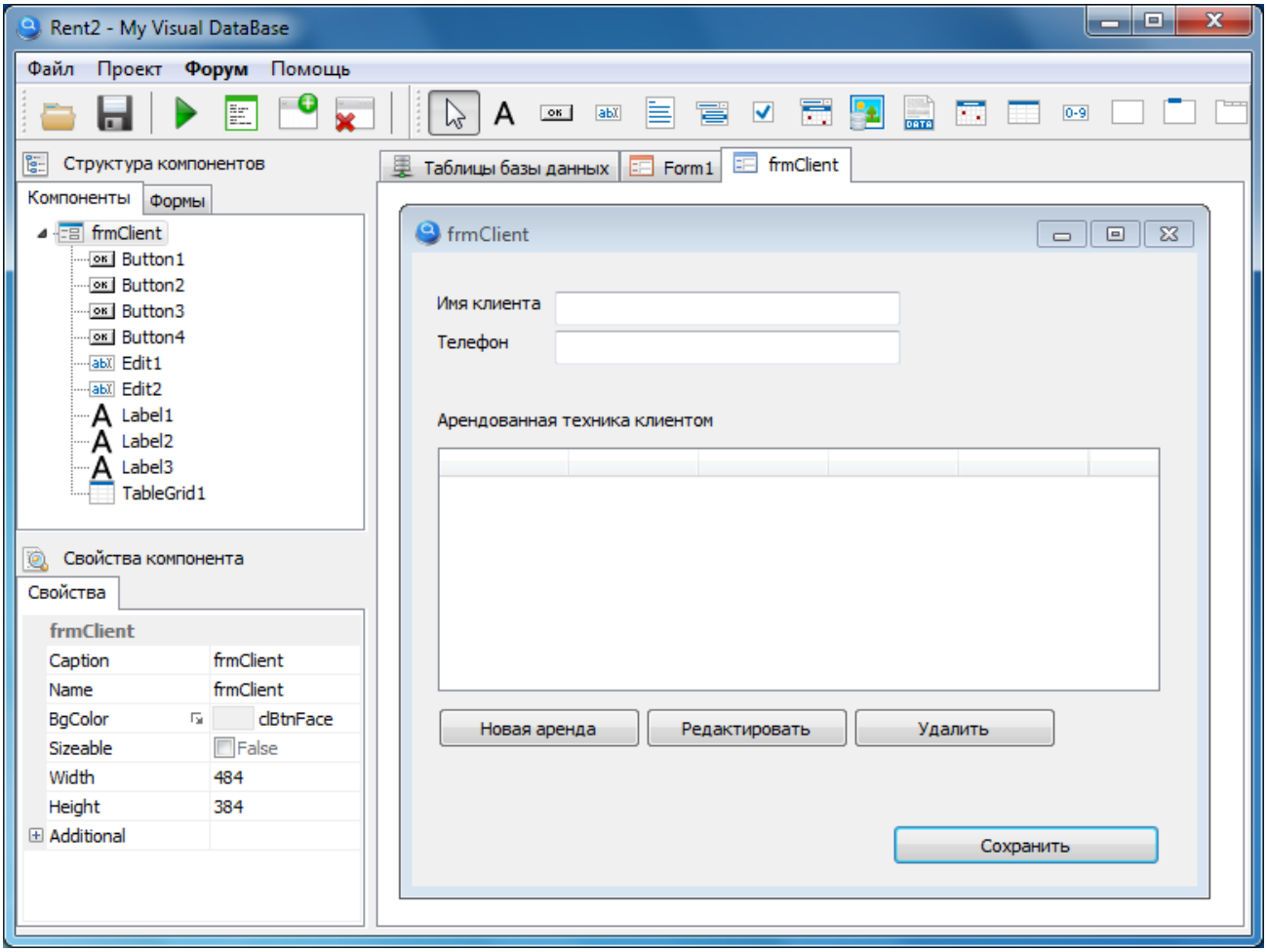


Рис.41

Именно данная форма является главным отличием от нашего предыдущего проекта. Теперь записи об аренде техники расположены не на главной форме, а на форме **frmClient**, и на этой форме вы видите не только данные клиента, такие как его имя и телефон, но и все записи связанные с арендой техники данным клиентом.

Форма **frmRent**:

Создайте новую форму, нажав кнопку  . Введите название формы **frmRent**

Расположите компоненты на данной форме как показано на рисунке 42.

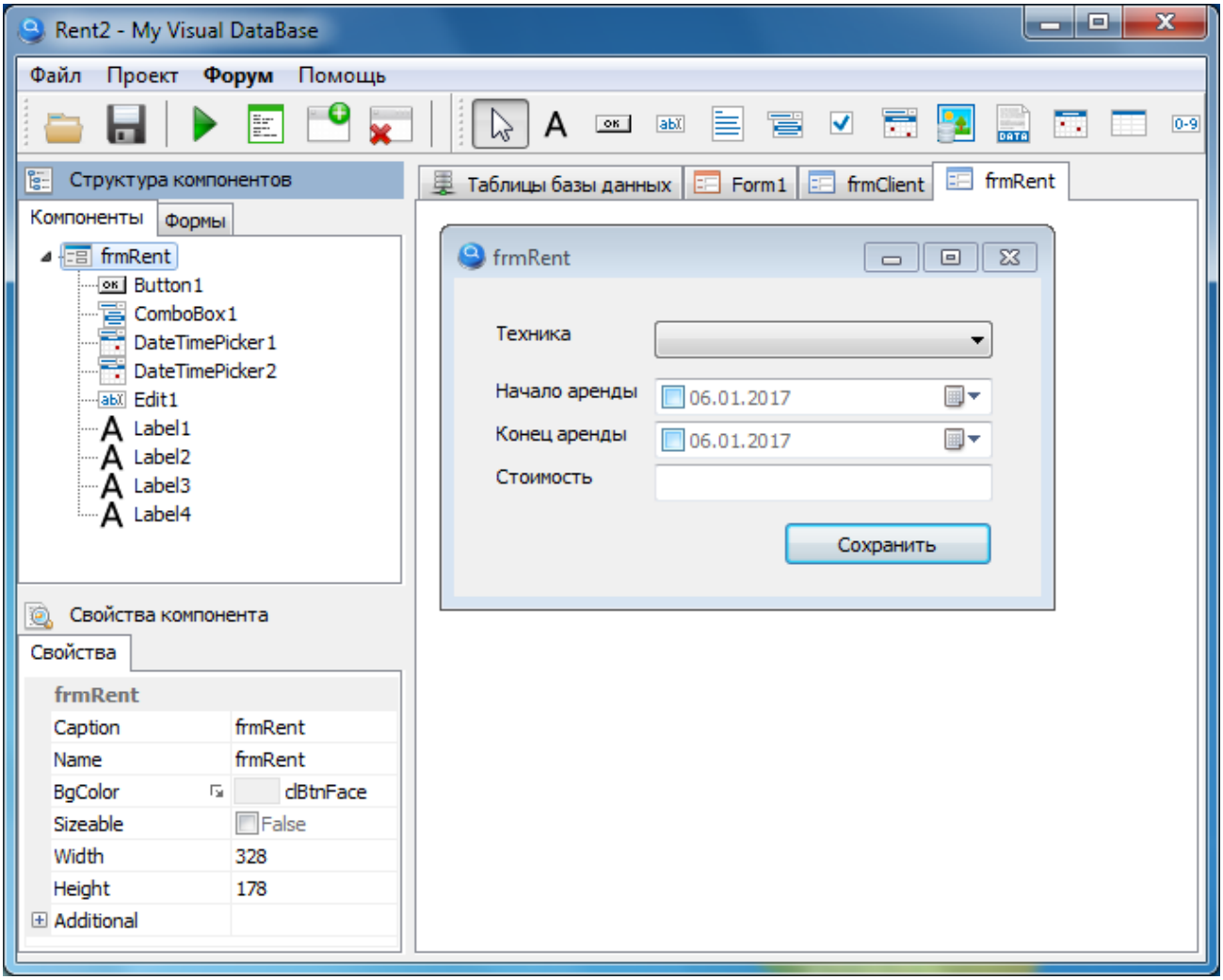


Рис.42

Данная форма предназначена для создания записи об аренде. Обратите внимание, в отличие от нашего предыдущего проекта, на данной форме нет выпадающего списка, в котором бы мы выбрали клиента, который арендует указанную технику.

Дело в том, что форма **frmRent** будет вызываться из формы **frmClient**, поэтому программа автоматически свяжет клиента с данной формой, и запись об арендованной технике автоматически будет принадлежать клиенту, данные которого будут на форме **frmClient**. Другими словами, вам просто не нужно беспокоиться об этом, программа автоматически свяжет клиента и арендованную им технику.

Форма **frmTechList**:

Создайте новую форму, нажав кнопку  . Введите название формы **frmTechList**

Расположите компоненты на данной форме как показано на рисунке 43.

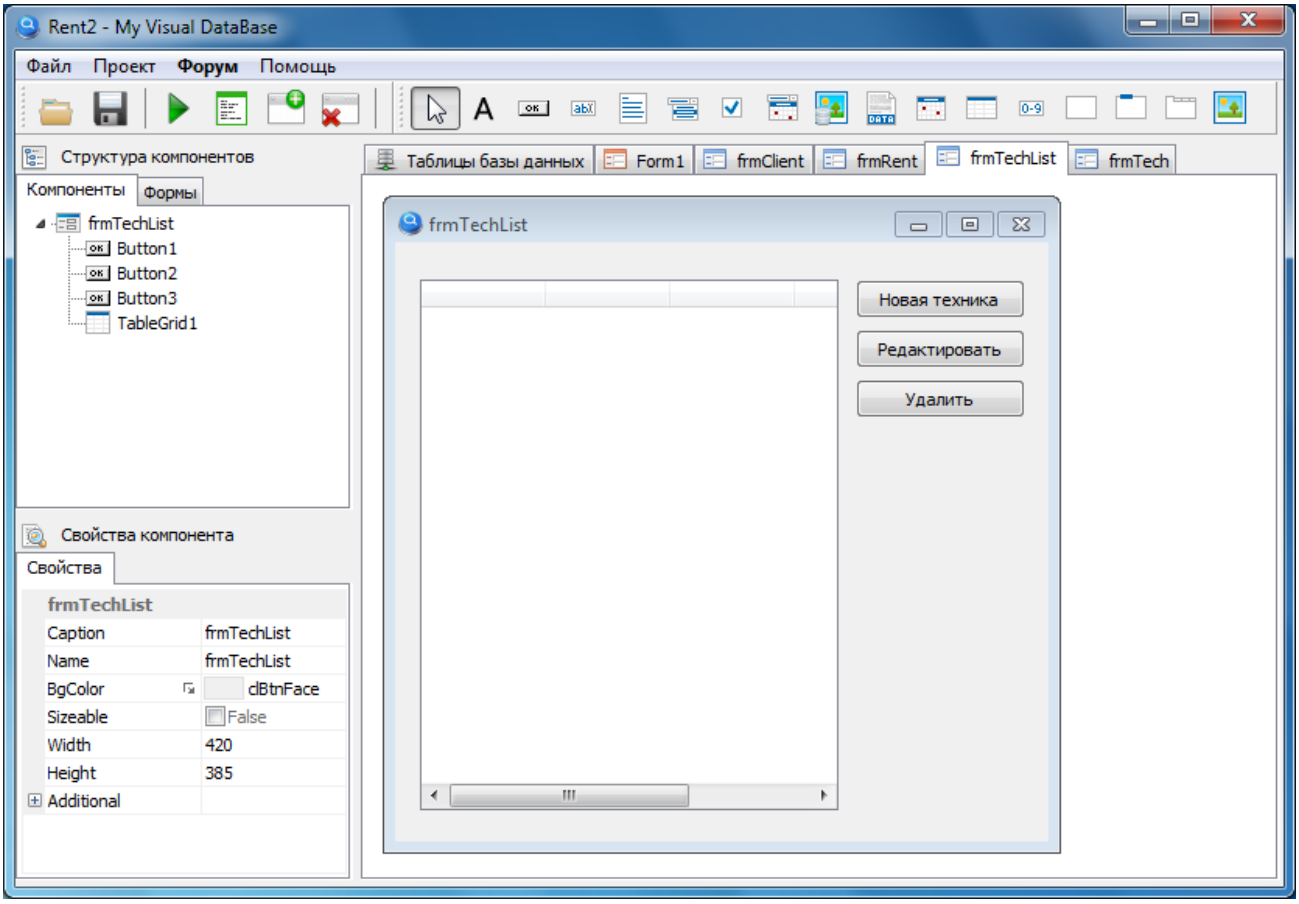


Рис.43

Данная форма точно такая же, как и в предыдущем проекте.

На создаваемой форме сможем видеть всю технику, которой располагаем, а так же вызвать форму для добавления новой техники, если вдруг такая появилась, либо редактировать информацию о существующей.

Форма **frmTech**:

Создайте новую форму, нажав кнопку  . Введите название формы **frmTech**

Расположите компоненты на данной форме как показано на рисунке 44.

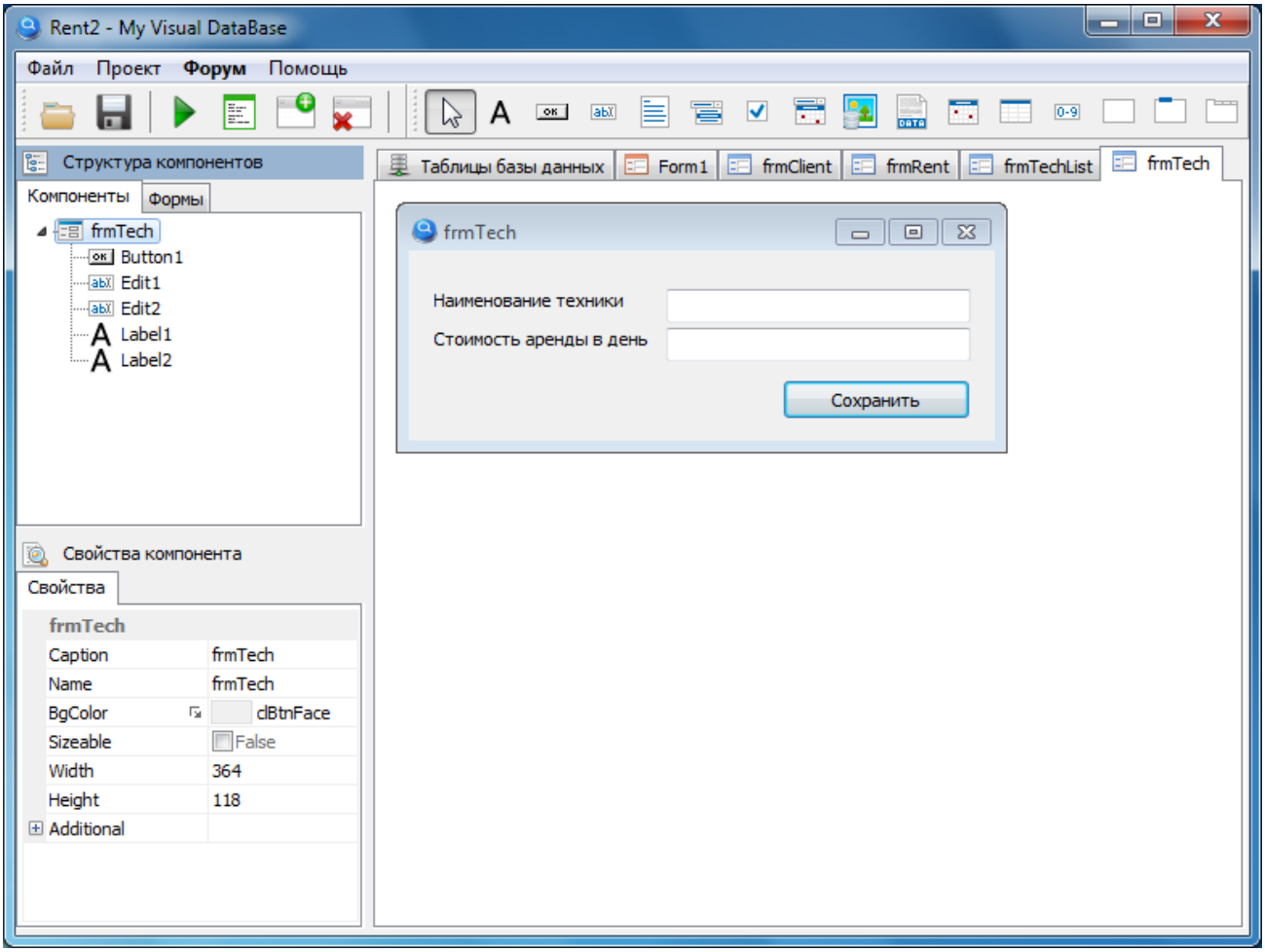


Рис.44

Данная форма точно такая же, как и в предыдущем проекте. Предназначена для создания/редактирования техники, которой вы располагаете, с указанием ее стоимости аренды в день.

Переходим к последнему этапу создания приложения, к настройке компонентов на форме.

Как вы помните, настройка компонентов сводится к тому, чтобы кнопкам  присвоить действие, а компонентам предназначенные для ввода информации (такие как:  и т.д.) указать, к какой таблице БД, и к какому полю они принадлежат.

Приступаем к настройке форм, я лишь перечислю, какие действия необходимо присвоить кнопкам и какие поля БД присвоить компонентам на форме. Подробней остановлюсь на компонентах, чья настройка отличается от предыдущего проекта.

Настройка формы **Form1**:

Имя клиента (текстовое поле)
TableName = Client
FieldName = Name

Телефон (текстовое поле)
TableName = Client
FieldName = Phone

Новый клиент (кнопка)
Действие = Новая запись
Форма = frmClient

Редактировать (кнопка)
Действие = Показать запись
Компонент таблицы = TableGrid1
Форма = frmClient

Удалить (кнопка)
Действие = Удалить запись
Компонент таблицы = TableGrid1

Чуть подробней остановимся на настройке кнопки **Найти**. Как я упомянул ранее, на главной форме теперь мы будем видеть записи о клиентах, а не записи об аренде, как в предыдущем проекте. Поэтому данная кнопка будет искать данные в таблице **Client**, вместо таблицы **Rent**.

На рисунке 45 можете видеть настройки кнопки **Найти**.

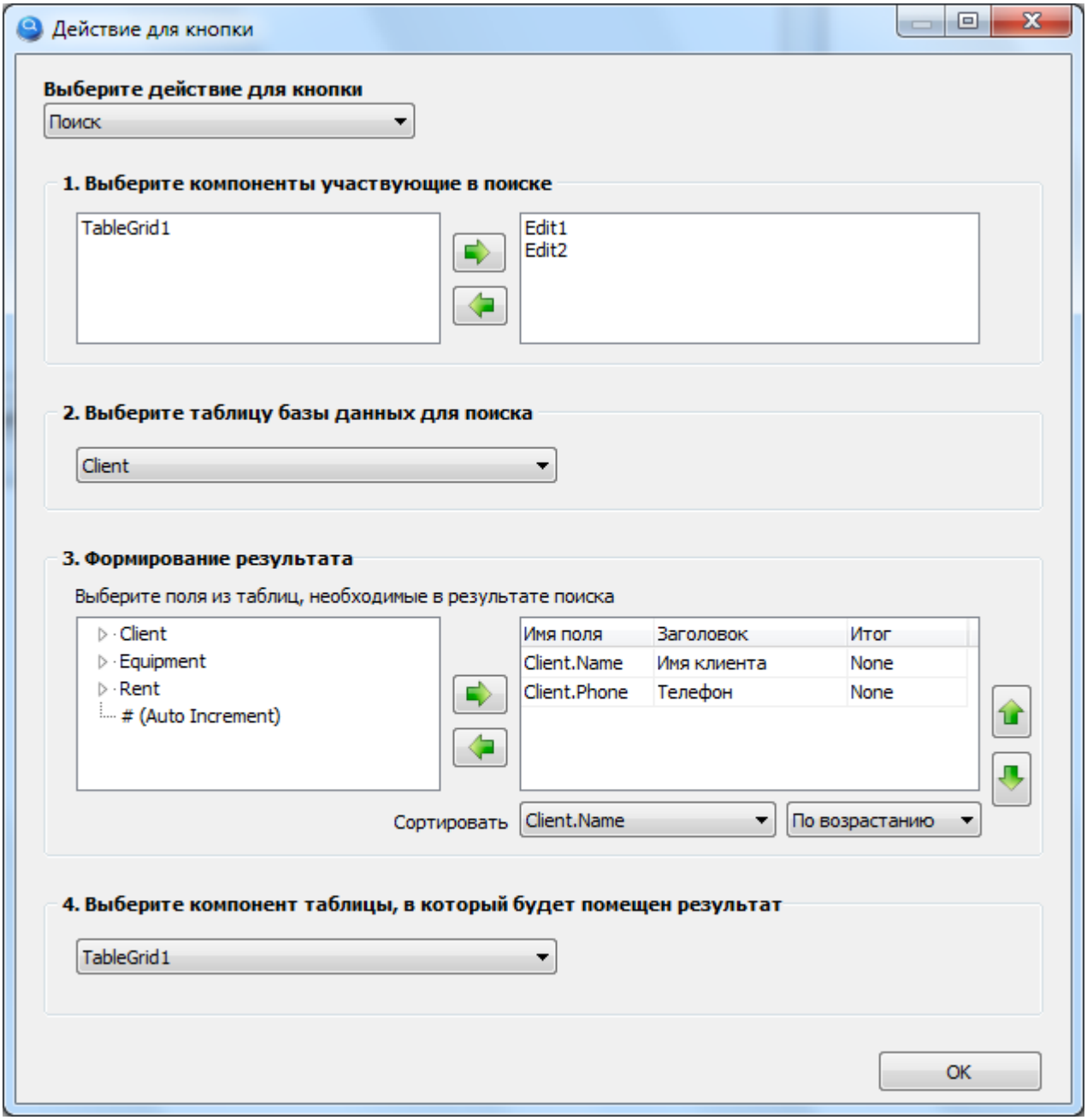


Рис.45

Настройка формы **frmClient**:

Имя клиента (текстовое поле)
TableName = Client
FieldName = Name

Телефон (текстовое поле)
TableName = Client
FieldName = Phone

Новая аренда (кнопка)
Действие = Новая запись
Форма = frmRent

Редактировать (кнопка)
Действие = Показать запись
Компонент таблицы = TableGrid1
Форма = frmRent

Удалить (кнопка)
Действие = Удалить запись
Компонент таблицы = TableGrid1

Чуть подробней остановимся на настройке кнопки **Сохранить.** Настройку данной кнопки можете видеть на рисунке 46.

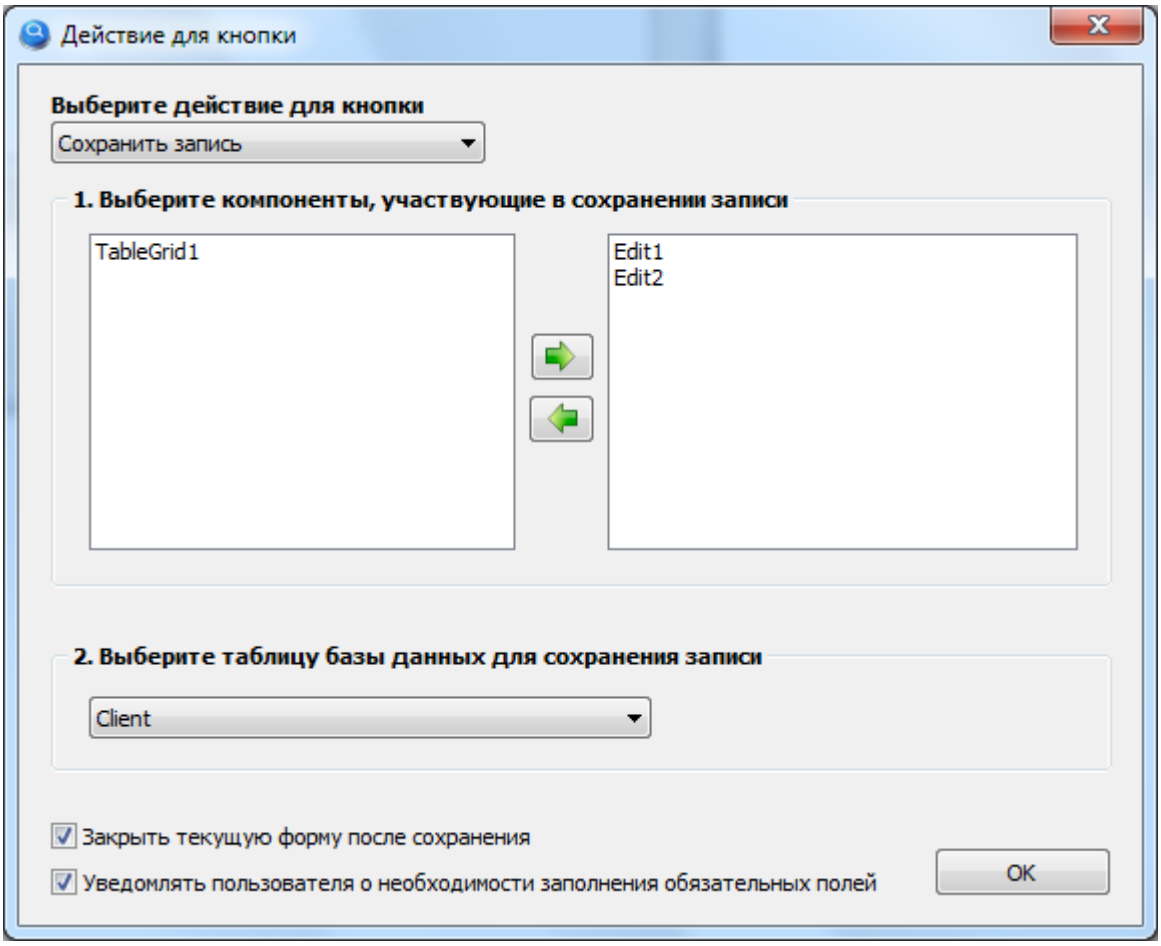


Рис.46

Обратите внимание, компонент **TableGrid1** не участвует в сохранении в записи, а просто показывает записи об аренде принадлежащие клиенту на форме. Поэтому оставьте его в левом списке.

Также подробней остановимся на настройке (рис.47) компонента **TableGrid1** (Арендованная техника клиентом). В данном компоненте мы будем видеть записи об аренде техники, которые принадлежат клиенту, чьи данные мы видим на текущей форме (**frmClient**).

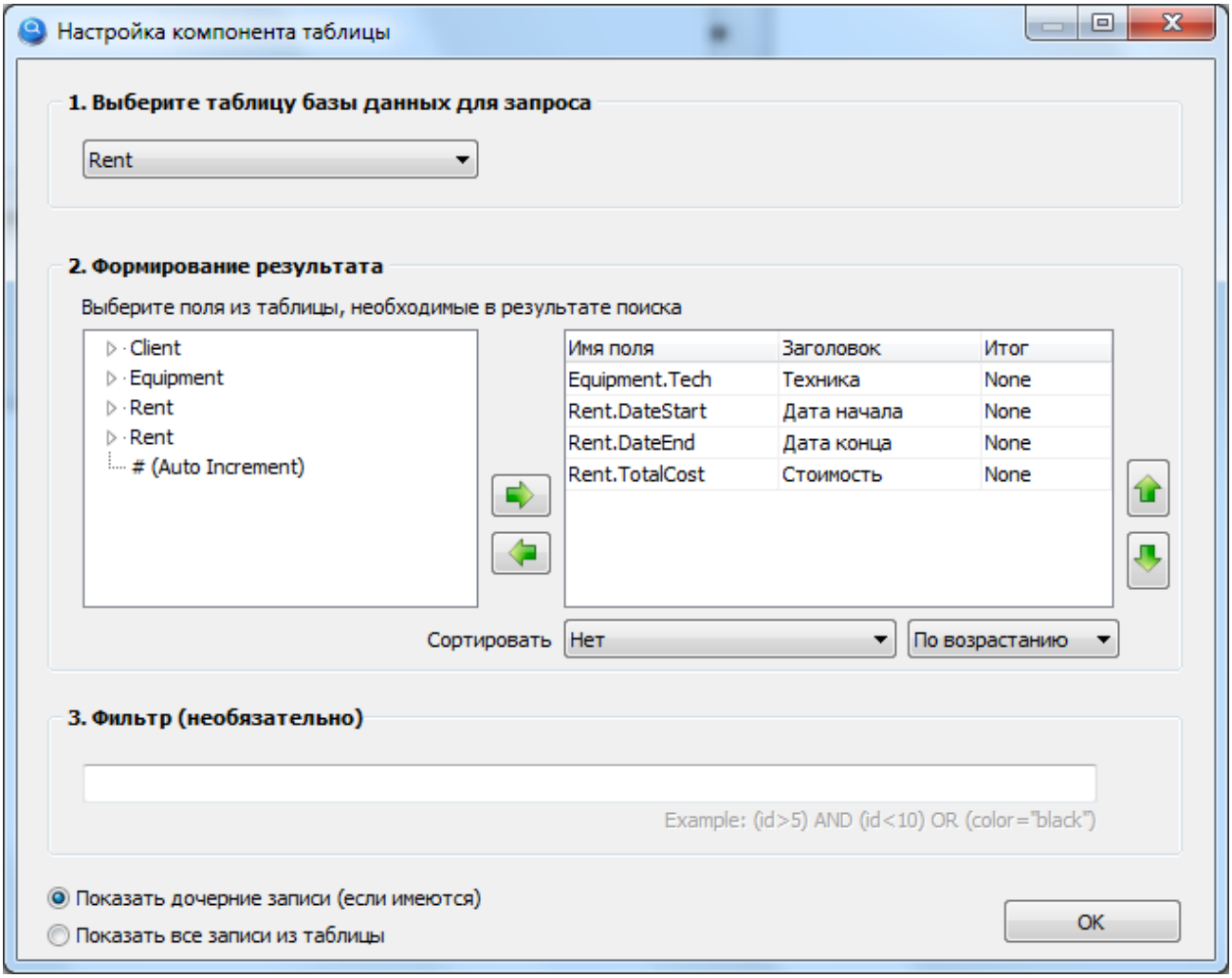
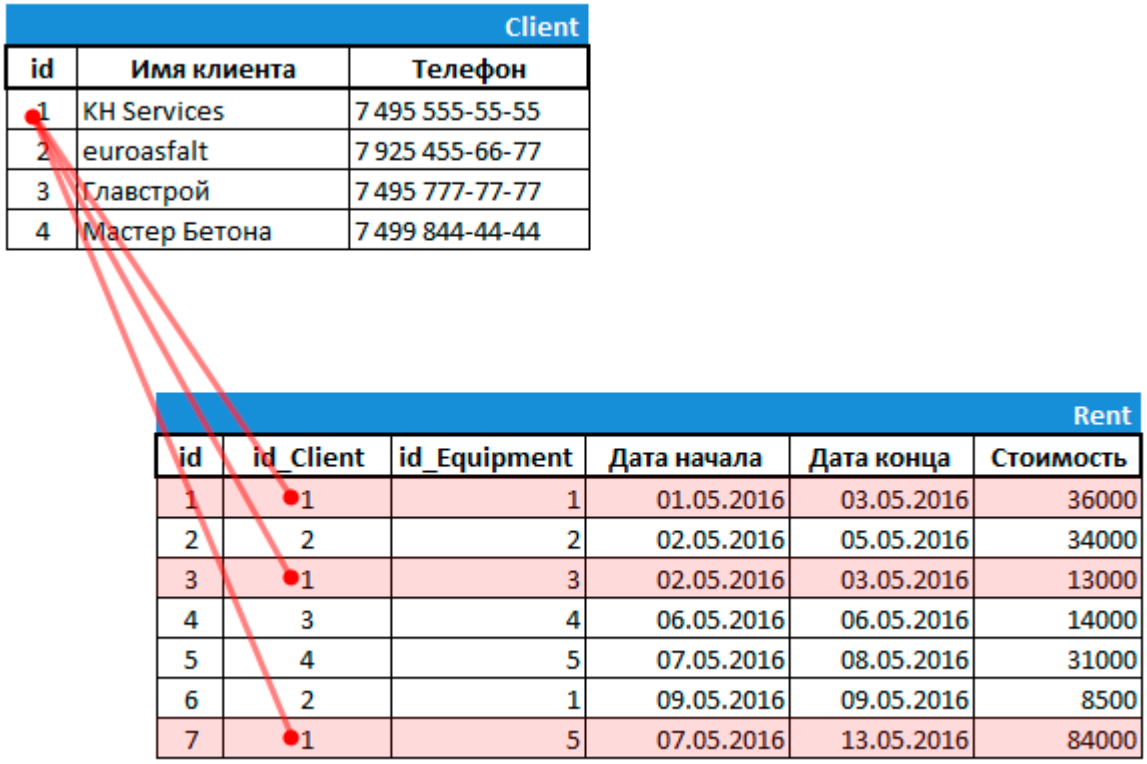


Рис.47

Обратите внимание на выбранную настройку **«Показать дочерние записи (если имеются)»**. Таким образом, в данном компоненте автоматически будут показаны дочерние записи, в нашем случае по отношению к клиенту, другими словами мы будем видеть записи об аренде, которые принадлежат клиенту.

Если кто-то забыл, что такое дочерние записи, можете взглянуть на рисунок 48.



Client

id	Имя клиента	Телефон
1	KH Services	7 495 555-55-55
2	euroasfalt	7 925 455-66-77
3	Главстрой	7 495 777-77-77
4	Мастер Бетона	7 499 844-44-44

Rent

id	id_Client	id_Equipment	Дата начала	Дата конца	Стоимость
1	1	1	01.05.2016	03.05.2016	36000
2	2	2	02.05.2016	05.05.2016	34000
3	1	3	02.05.2016	03.05.2016	13000
4	3	4	06.05.2016	06.05.2016	14000
5	4	5	07.05.2016	08.05.2016	31000
6	2	1	09.05.2016	09.05.2016	8500
7	1	5	07.05.2016	13.05.2016	84000

Рис.48

Т.е. если на форме **frmClient** мы видим данные о клиенте **KH Services**, то в компоненте **TableGrid1** мы увидим дочерние записи из таблицы Rent, которые на рисунке отмечены красным.

Настройка остальных форм, такие как frmRent, frmTechList, frmTech ничем не отличается от нашего предыдущего проекта, настройте их самостоятельно.

После того, как вы закончили с настройками оставшихся форм, время запустить наш проект, нажав кнопку  и протестировать его работу.

Как и в прошлом проекте, сначала необходимо добавить в базу данных технику, которой мы располагаем и ее стоимость за 1 день аренды, для этого на главной форме нажмите кнопку «Техника» (рис.49).

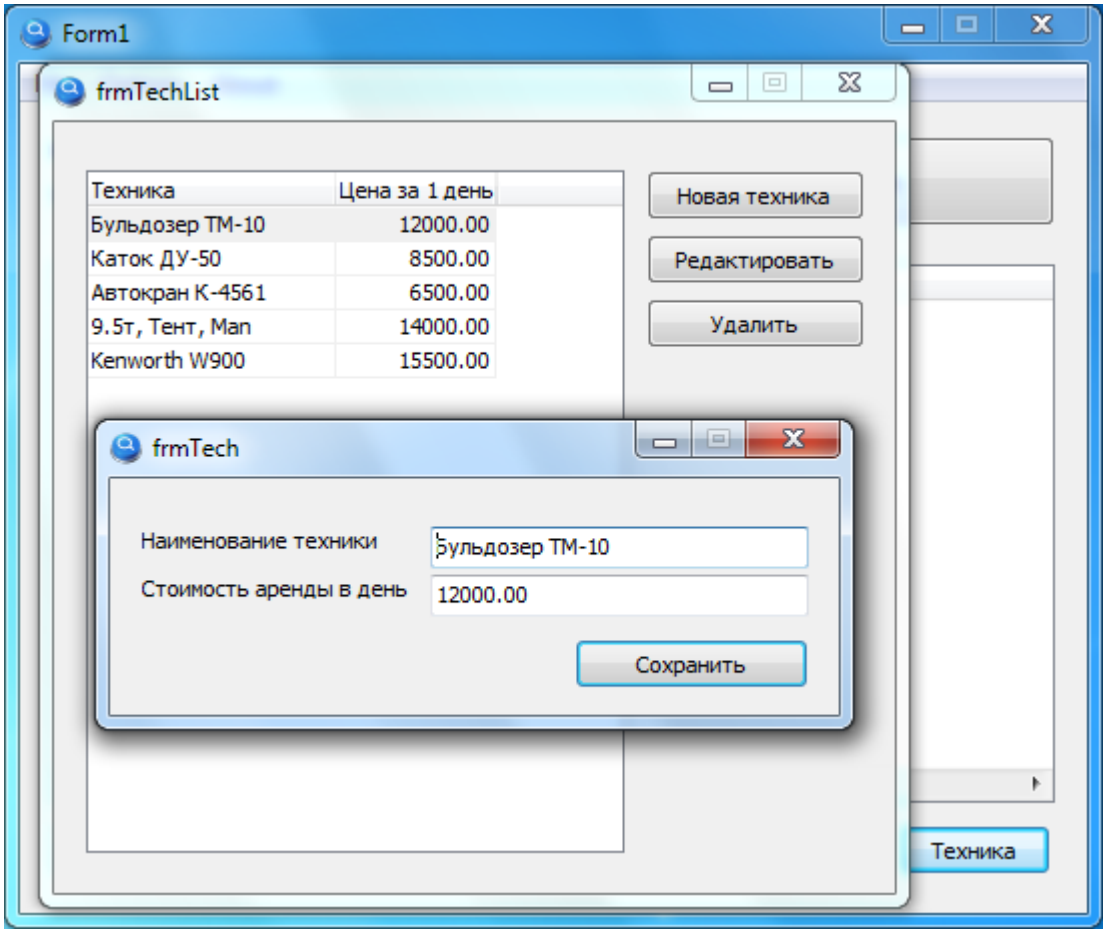


Рис.49

После того, как вы добавил в базу данных всю технику, который располагаете, можно приступить непосредственно к работе.

Допустим, к нам обратилась фирма «KH Services», чтобы арендовать Бульдозер ТМ-10, Автокран К-4561 и тягач Kenworth W900. И если эта фирма обратилась к нам первый раз, тогда на главной форме нажимаем кнопку «Новый клиент». Появится форма **frmClient** на которой мы введем данные клиента и на этой же форме нажмем кнопку **«Новая аренда»**, чтобы поочередно присвоить данному клиенту арендованную им технику (рис.50).

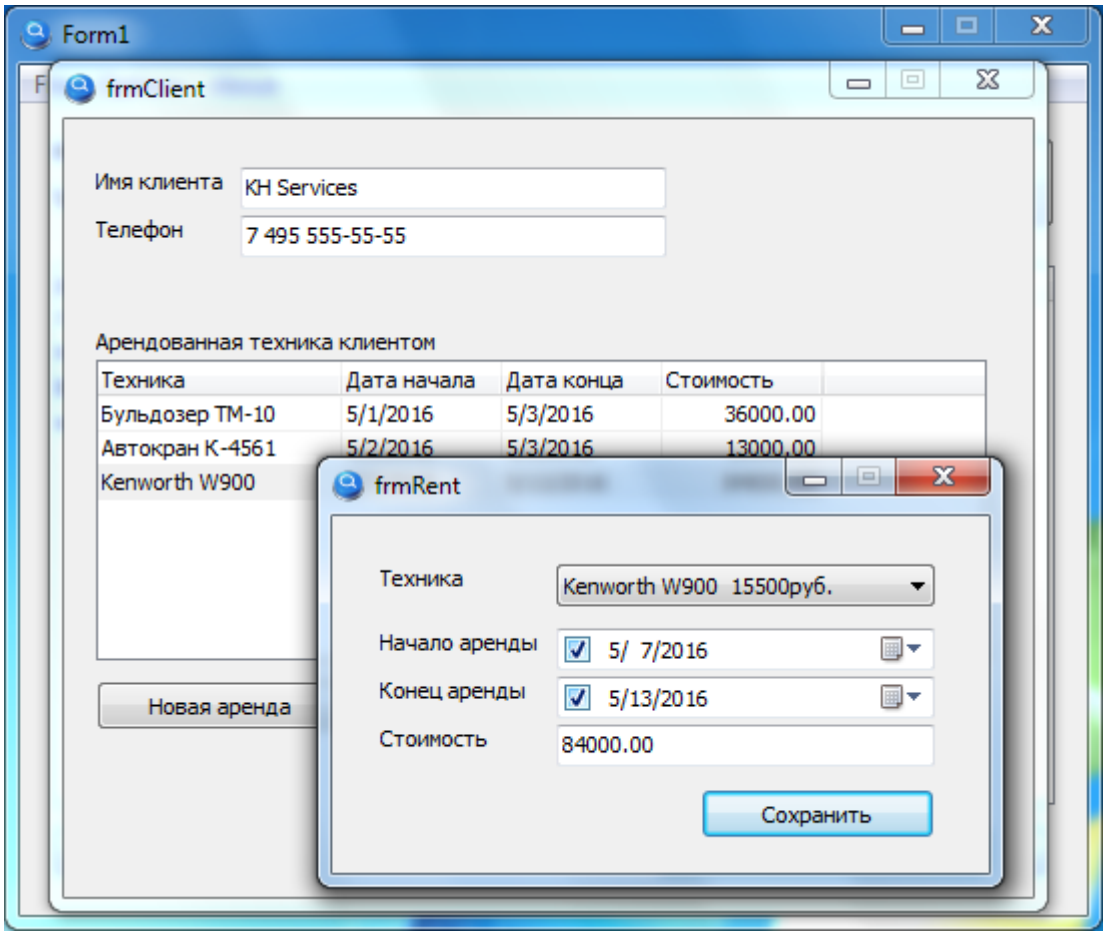


Рис.50

Обратите внимание, если клиент уже существует в базе данных, вы **не** должны создавать его снова. Вместо этого вы должны найти его на главной форме и нажать кнопку «Редактировать», что вызовет форму **frmClient**, на которой с помощью кнопки «Новая аренда» точно также присвоите данному клиенту очередную арендованную им технику.

На главной форме с помощью поля «Имя клиента» вы можете произвести поиск по базе данных, чтобы найти клиента. По умолчанию поиск происходит по полному совпадению имени, т.е. чтобы найти клиента с именем KH Services, вам необходимо в текстовом поле также полностью ввести его имя, что не всегда бывает удобным. Поэтому в настройках данного текстового поля вы можете изменить режим поиска на частичный, таким образом, вы найдете клиента KH Services даже если в качестве поиска введете просто service.

Найдите свойство Filter и выберите его значение %s% как показано на рисунке 51.

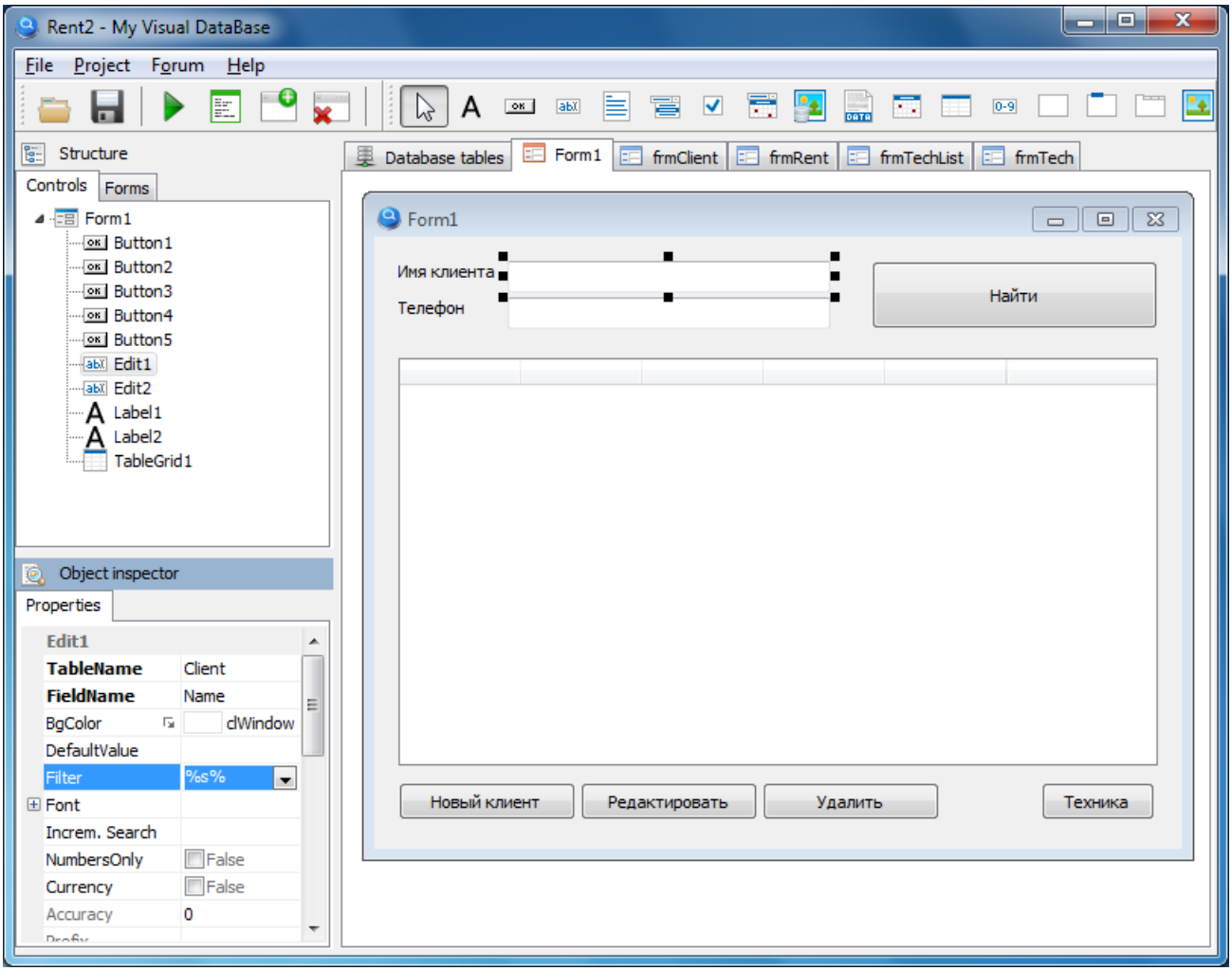


Рис.51

2.2 Нормализация структуры базы данных.

Давайте отдохнем от практики и немного углубимся в теорию, которая поможет нам избежать ошибок, связанных с проектированием структуры базы данных.

Поговорим о нормализации базы данных. Если кто-то предпочитает объяснение в академическом стиле, могут подчерпнуть информацию об этом Wikipedia: https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0

Я же постараюсь объяснить принципы нормализации более простым языком на примерах.

Во-первых, нужно понять, зачем нам заниматься нормализацией структуры базы данных.

Нормализация это набор правил, которых мы должны придерживаться при создании структуры базы данных. Это поможет нам избежать ошибок в структуре БД, которые могут привести к избыточным или даже противоречивым данным.

Сами того не подозревая, но в начале книги мы уже занимались нормализацией, когда рассматривали проблемы хранения данных в одной таблице, и разделяли ее на несколько более маленьких таблиц.

Давайте представим, что мы снова ничего не знаем о правилах проектирования баз данных и создали таблицу БД с подобной структурой (рис.52):

База данных персоонала							
id	**Имя**	**Фамилия**	**Хобби**	**Город**	**Индекс**	**Зарплата**	**Налог**
1	Иван	Иванов	Футбол, Тенис, Музыка	Ярославль	150000	25000	3250
2	Петр	Петров	Танцы	Тверь	170000	23000	2990
3	Михаил	Сидоров	Шахматы, Чтение	Тула	300000	28000	3640

Рис.52

Приступим к нормализации.

Первая нормальная форма

Определение первой нормальной формы академическим языком звучит так:

> *Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов*

Если говорить простым языком, то в поле не должно быть несколько значений, одно поле = одно значение. В нашем примере обратите внимание на колонку с именем «Хобби», в котором через запятую указаны хобби человека. Это не допустимо, просто запомните это.

Чтобы таблица не нарушала это правило, нам придется создать дополнительные записи для каждого хобби человека, т.е. дублируем данные, тем самым намеренно создаем избыточность данных в таблице (рис.53).

База данных персоонала							
id	**Имя**	**Фамилия**	**Хобби**	**Город**	**Индекс**	**Зарплата**	**Налог**
1	Иван	Иванов	Футбол	Ярославль	150000	25000	3250
2	Иван	Иванов	Тенис	Ярославль	150000	25000	3250
3	Иван	Иванов	Музыка	Ярославль	150000	25000	3250
4	Петр	Петров	Танцы	Тверь	170000	23000	2990
5	Михаил	Сидоров	Шахматы	Тула	300000	28000	3640
6	Михаил	Сидоров	Чтение	Тула	300000	28000	3640

Рис.53

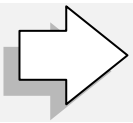
Избавиться от избыточности данных нам поможет правило второй нормальной формы.

Вторая нормальная форма

Определение второй нормальной формы академическим языком звучит так:

> *Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа.*

Если простым языком, то у каждой таблицы должен быть первичный ключ, который однозначно идентифицирует запись в таблице. В нашем случае это столбец с именем id, в котором хранится идентификатор записи, он уникален для каждой записи в таблице.



Если вы используете программу My Visual Database, то при создании таблицы БД, данный первичный ключ всегда создается автоматически. Вам просто не нужно беспокоиться об этом.

Также это правило говорит о том, что в таблице не должно быть дублирующих данных (избыточность). Если вы посмотрите на таблицу (рис.53), которая находится в первой нормальной форме, то увидите, что некоторые записи содержат одинаковые данные, т.е. данные дублируются. Причиной тому столбец «Хобби».

Чтобы решить проблему с избыточностью данных, необходимо создать еще одну таблицу, в которой будем хранить хобби персонала, также в таблице будет присутствовать внешний ключ, который будет определять, какому человеку принадлежит указанное хобби.

Теперь, когда у нас две таблицы, мы можем создать любое количество хобби, которое может принадлежать человеку, тем самым избежав избыточности данных. Чтобы совсем стало ясно, взгляните на рисунок 54.

Persons

id	Имя	Фамилия	Город	Индекс	Зарплата	Налог
1	Иван	Иванов	Ярославль	150000	36000	3250
2	Петр	Петров	Тверь	170000	34000	2990
3	Михаил	Сидоров	Тула	300000	13000	3640

Hobby

id	id_Persons	Хобби
1	1	Футбол
2	1	Тенис
3	1	Музыка
4	2	Танцы
5	3	Шахматы
6	3	Чтение

Рис.54

Третья нормальная форма.

Определение третьей нормальной формы академическим языком звучит так:

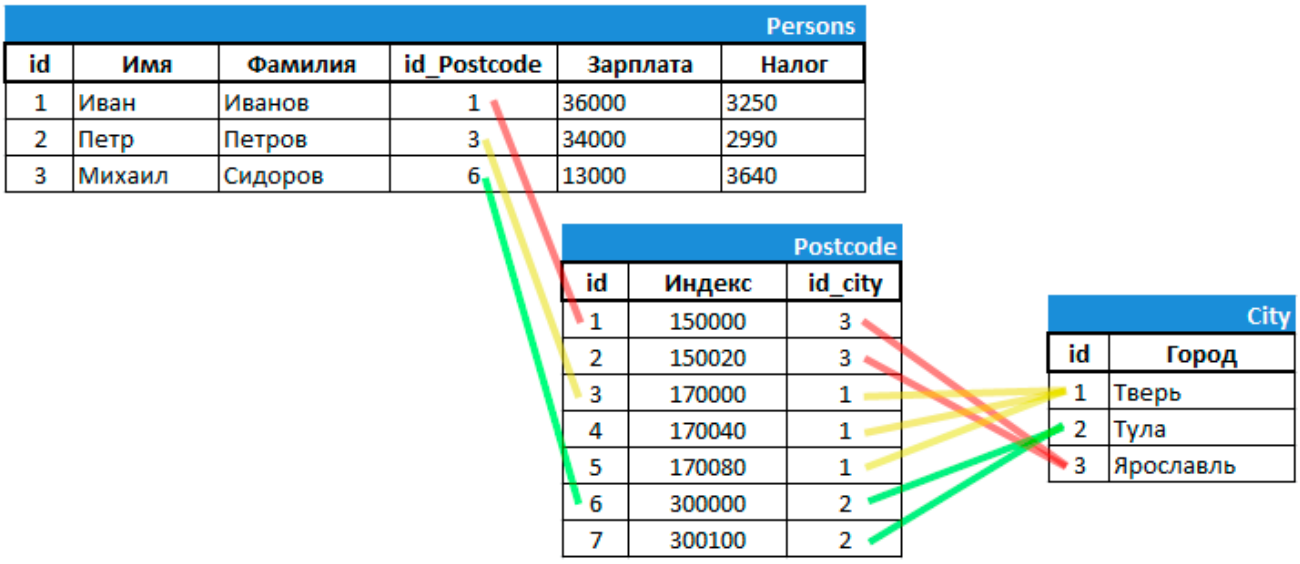
> *Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.*

Третья нормальная форма необходима, чтобы бороться с так называемой **транзитивной зависимостью**.

Для начала разберемся, что это. Обратите внимание, что в таблице персонала (рис.54) имеются столбцы **Город** и **Индекс**. К сожалению, людям свойственно ошибаться. Допустим, человек наполняя базу данных, ошибся и ввел сотруднику, живущему в Ярославле Тверской индекс, чему после этого верить? Как результат мы получим противоречивые данные.

Таким образом, столбцы **Город** и **Индекс** имеют зависимость друг от друга. Но ведь если мы знаем Индекс, мы можем узнать и город по этому индексу. Поэтому, почему бы нам в таблице персонала не хранить только **Индекс**?

Теперь чтобы избавиться от этой транзитивной зависимости, нам необходимо создать еще две таблицы. Таблицу с названиями городов и таблицу с индексами, в которой также будет внешний ключ, определяющий, какому городу принадлежит индекс (рис.55).



Persons

id	Имя	Фамилия	id_Postcode	Зарплата	Налог
1	Иван	Иванов	1	36000	3250
2	Петр	Петров	3	34000	2990
3	Михаил	Сидоров	6	13000	3640

Postcode

id	Индекс	id_city
1	150000	3
2	150020	3
3	170000	1
4	170040	1
5	170080	1
6	300000	2
7	300100	2

City

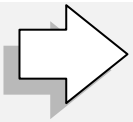
id	Город
1	Тверь
2	Тула
3	Ярославль

Рис.55

На практике же не всегда пользуются третьей нормальной формой, оставляя поля с транзитивной зависимостью.

Если вы проектируете базу данных, которая будет использоваться, например, в качестве личного справочника, можете пойти на компромисс и оставить все как есть, жертвуя возможным появлением противоречивых данных в угоду простоты. Но если вы проектируете базу данных для использования, например в банковском секторе, на вашем месте я бы не игнорировал третью нормальную форму.

Давайте снова взглянем на рисунок 55, так как в таблице **Persons** есть еще два поля с транзитивной зависимостью. Это поля **Зарплата** и **Налог**. Допустим у нас фиксированный налог 13%, поэтому зная **зарплату**, всегда можно высчитать и **налог**. Обычно вы не должны хранить данные в таблице, которые могут быть получены из других полей таблицы, поэтому поле **Налог** можете смело удалить.



Как только вы поймете эти правила, они вам покажутся вполне естественными. При дальнейшем проектировании баз данных, вы сами того не заметите как будете придерживаться их совершенно инстинктивно.

2.3. Каскадное удаление и поддержка целостности.

Надеюсь, вы не забыли про наш совместный с вами проект по аренде строительной техники. Мы и дальше будем работать с ним, расширяя его функционал, тем самым узнавая, что то новое о базах данных.

Запустите My Visual Database и откройте снова наш проект, который находится в папке «Rent project2» .

Откройте вкладку «Таблицы базы данных», чтобы видеть созданные таблицы в нашей БД (рис. 56).

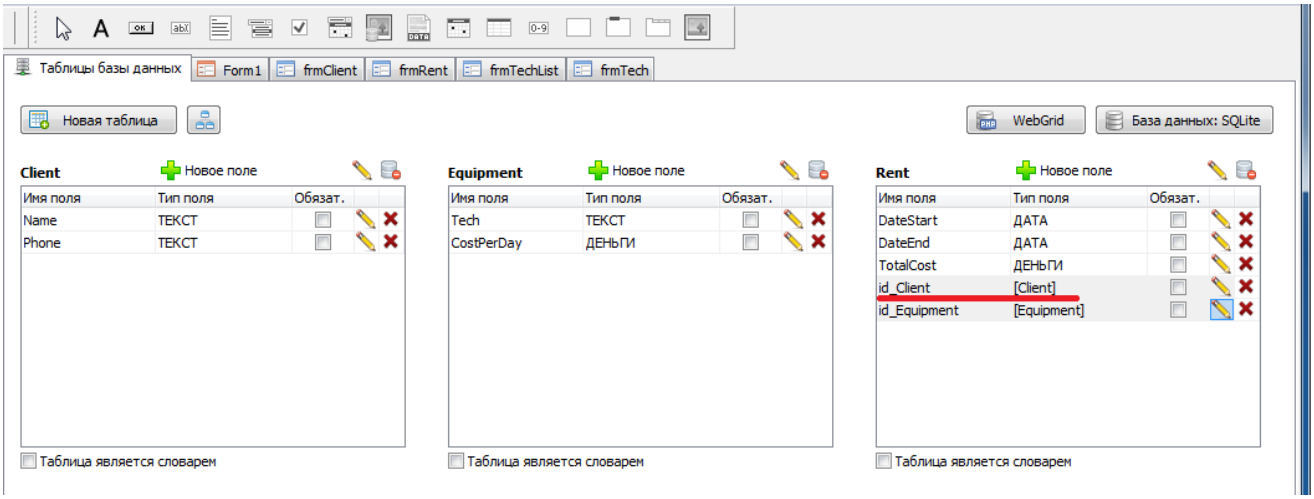


Рис.56

И поговорим про каскадное удаление.

Бывает так, что нам необходимо удалить клиента из базы данных, и все данные, которые связаны с ним. В нашем случае, при удалении клиента (таблица Client) необходимо удалить и все данные об аренде техники данным клиентом (таблица Rent), иначе мы получим записи об аренде, которые ссылаются на несуществующего клиента, что бывает недопустимым.

Обратите внимание на внешний ключ **id_Client** в таблице **Rent**, который ссылается на клиента из таблицы **Client**. Для данного внешнего ключа необходимо задействовать так называемое **каскадное удаление**. После чего, при удалении клиента из таблицы **Client**, автоматически будут удаляться и записи из таблицы **Rent**, которые ссылаются на удаляемого клиента благодаря внешнему ключу **id_Client**.

Другими словами, при удалении родительской записи, будут удалены и дочерние записи.

Задействовать каскадное удаление для внешнего ключа можно так, как показано на рисунке 57.

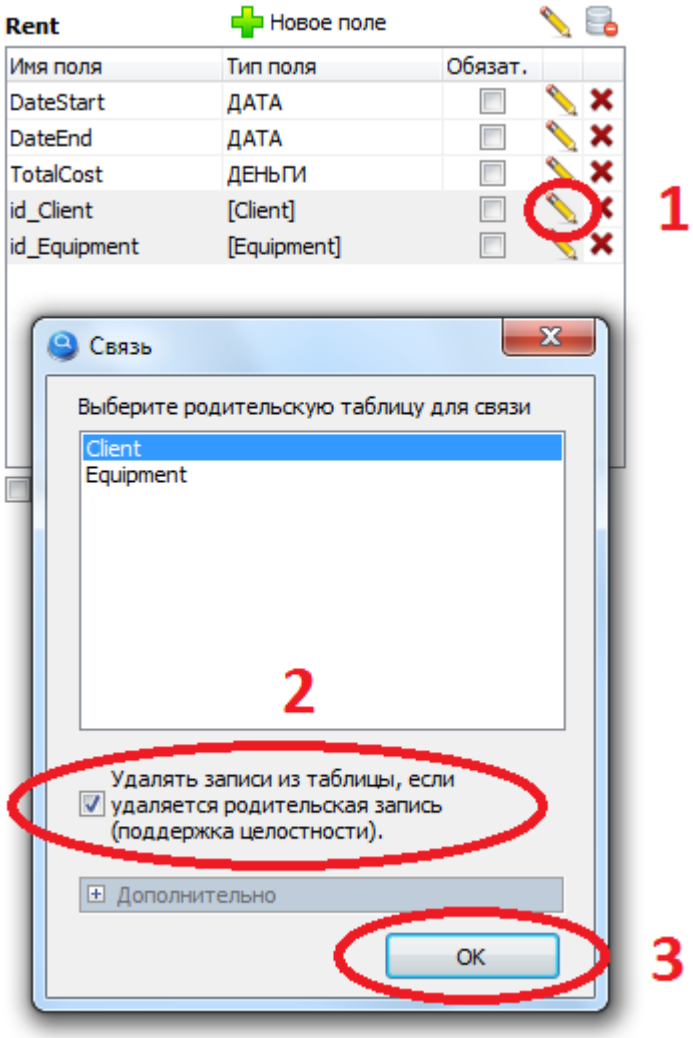


Рис.57

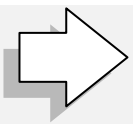
Честно говоря, без каскадного удаления, вы бы и не смогли удалить клиента из таблицы **Client**, потому что данный клиент имеет записи об аренде техники. Другими словами, потому что в таблице **Rent** имеются записи, в которых внешний ключ **id_Client** ссылается на клиента.

Но если вдруг клиент пока не имеет записей об аренде, база данных позволит его удалить, т.к. при этом не нарушается целостность данных в других таблицах.

То же самое можно сказать и про внешний ключ **id_Equipment** в таблице **Rent**,вы не сможете удалить оборудование из таблицы **Equipment**, если на данное оборудование ссылается хотя бы одна запись в других таблицах.

Таким образом, база данных автоматически поддерживает свою целостность, чтобы не допустить ситуацию, когда внешний ключ ссылается на несуществующую запись.

Поэтому, либо вы удаляете запись и все связанные с этой записью данные из БД, используя каскадное удаление, либо не удаляете вовсе.



Как правило не принято из базы данных что либо удалять, вместо этого рекомендуется помечать запись как Архив, используя поле с типом «Да/Нет».

2.4. Добавление типа клиента.

Продолжаем нашу практику.

Давайте сделаем так, чтобы можно было присвоить клиенту его тип, например, является ли он «Физическое лицо» либо «Юридическое лицо».

И тут вас может подстерегать ошибка. Есть большой соблазн создать новое текстовое поле в таблице БД «Client», куда бы вы просто вписывали тип клиента. В результате, ваши данные в таблице «Client» выглядели бы, например, так как показано на рисунке 58.

Client

Имя клиента	Тип клиента	Телефон
KH Services	Юридическое лицо	7 495 555-55-55
euroasfalt	Юридическое лицо	7 925 455-66-77
Главстрой	Юридическое лицо	7 495 777-77-77
Мастер Бетона	Юридическое лицо	7 499 844-44-44
Иванов Иван Иваныч	Физическое лицо	7 495 989-98-98
Петров Петр Петрович	Физическое лицо	7 910 933-01-02

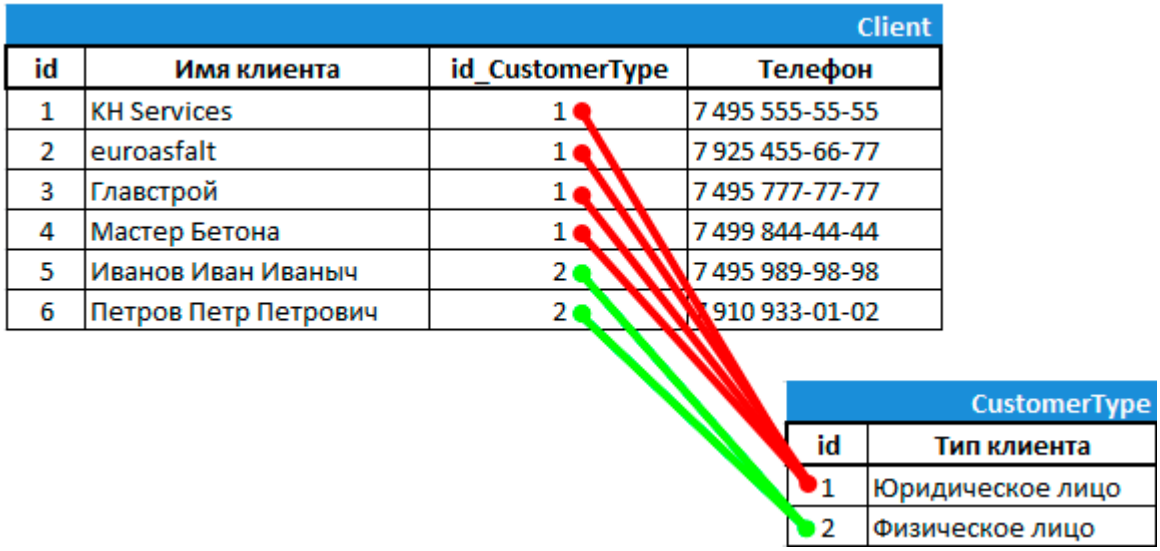
Рис.58

Обратите внимание на колонку «Тип клиента», по задумке в этой колонке у нас может быть только два значения, либо «Юридическое лицо», либо «Физическое лицо», таким образом, мы наблюдаем явную **избыточность данных**, что противоречит второй нормальной форме (глава 2.2).

Нет никакого смысла для каждого клиента, вручную, с помощью текста писать его тип. Это недопустимо при проектировании баз данных, т.к. впоследствии это скажется на ее производительности, а также на большую вероятность появления ошибок в БД, например кто-то решит, что достаточно написать просто «Юр. Лицо», или «Ю.Л.», тем самым потеряется целостность данных и сильно усложнится работа с ними.

Так как же правильно?

Правильным решением, будет создать еще одну таблицу, в которой будут перечислены все допустимые типы клиентов, назовем её «CustomerType». А в таблице «Client» создать внешний ключ, который будет ссылаться таблицу «CutomerType», таким образом, во внешнем ключе будет сохраняться идентификатор выбранного типа клиента. Чтобы стало совсем ясно, взгляните на рисунок 59.



Client

id	Имя клиента	id_CustomerType	Телефон
1	KH Services	1	7 495 555-55-55
2	euroasfalt	1	7 925 455-66-77
3	Главстрой	1	7 495 777-77-77
4	Мастер Бетона	1	7 499 844-44-44
5	Иванов Иван Иваныч	2	7 495 989-98-98
6	Петров Петр Петрович	2	7 910 933-01-02

CustomerType

id	Тип клиента
1	Юридическое лицо
2	Физическое лицо

Рис.59

При таком подходе, вам не придется каждый раз вручную писать тип клиента, вы просто выберите его из списка. Более того, вы можете добавить в таблицу «CustomerType» новые типы клиентов, например, «Индивидуальный предприниматель», «Гос. предприятие» и т.д.

Также ничто не мешает вам переименовывать типы клиентов и это никак не скажется на целостности данных, т.к. во внешнем ключе «id_CustomerType» таблицы «Client» сохраняется лишь числовой идентификатор типа клиента, а не его текстовое представление.

С теоретической частью разобрались, теперь опробуем это на практике:

1. Как было сказано выше, нам нужно создать еще одну таблицу в базе данных, назовем ее «CustomerType».
2. В этой таблице необходимо создать колонку с типом ТЕКСТ, назовем ее «TypeName».
3. В таблице «Client», создайте связь с таблицей «CustomerType»

В результате у вас должно получиться так, как показано на рисунке 60.

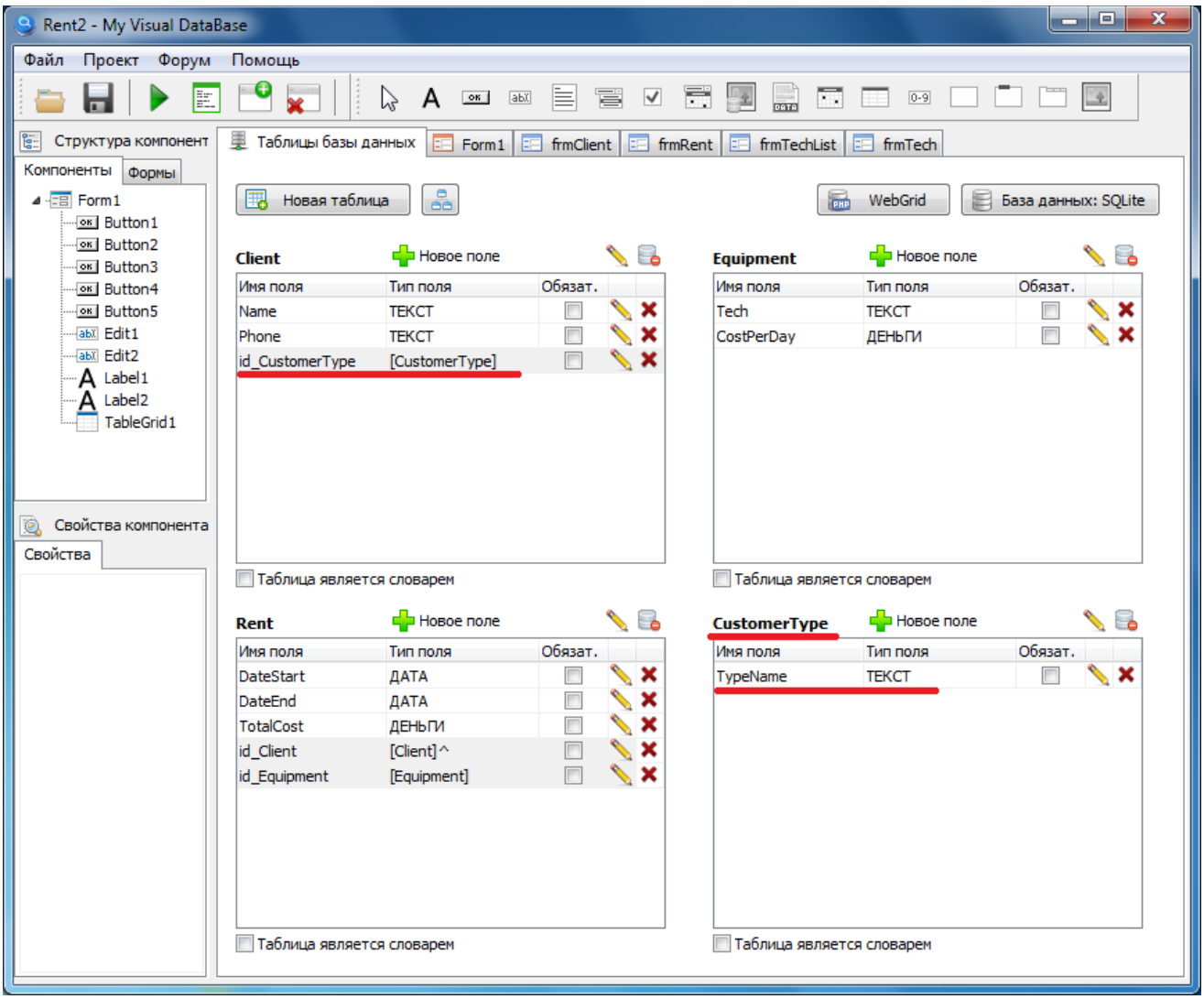


Рис.60

Посмотрите на формы **frmTechList** и **frmTech**. Точно такие же необходимо создать для работы с типами клиента. Первая для того, чтобы показать список добавленных записей и вторую для создания/редактирования записей.

Создайте новую форму с именем **frmCustTypeList** и расположите на ней компонент TableGrid и три кнопки, как показано на рисунке 61.

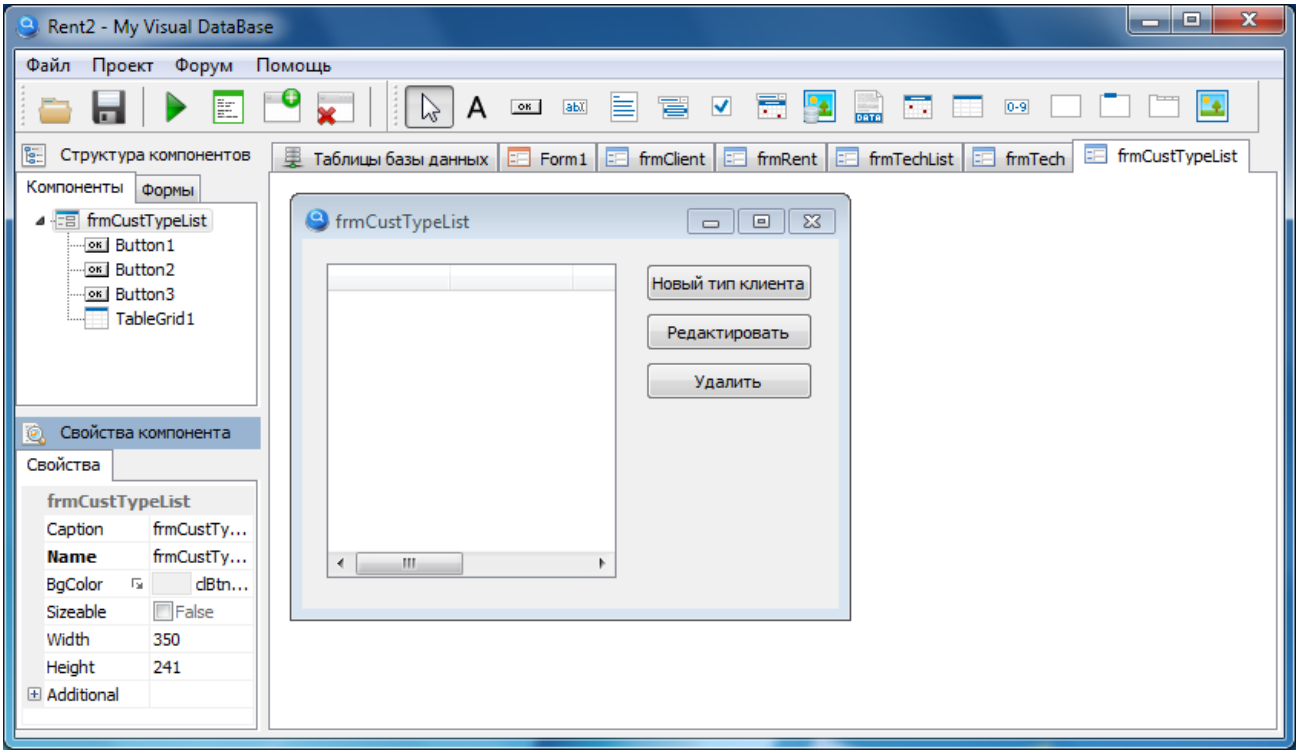


Рис.61

Также создайте форму с именем **frmCustTypeAdd**, как показано на рисунке 62.

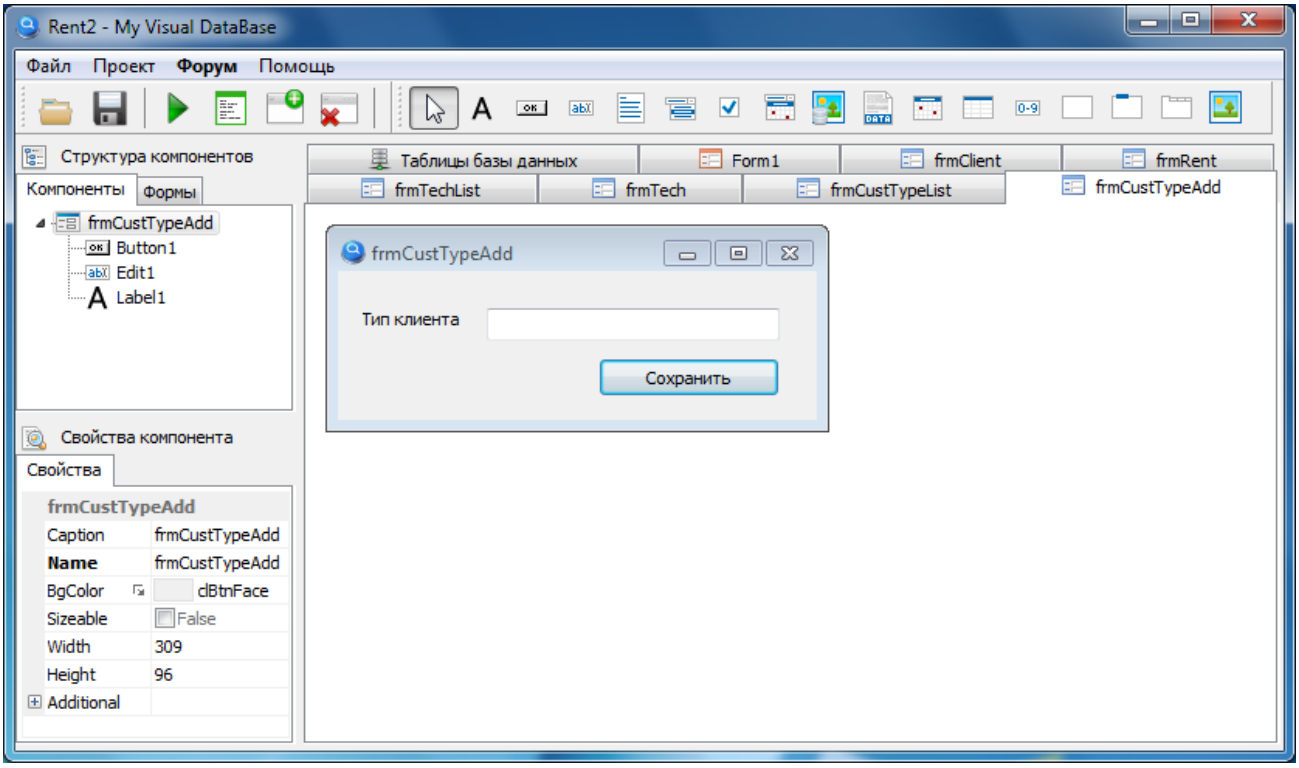


Рис.62

Теперь осталось настроить компоненты на созданных формах. Настройка как обычно довольно проста.

Настройка формы **frmCustTypeList**:

TableGrid1

1. Зайдите в настройки компонента TableGrid1, чтобы настроить его на показ данных из таблицы БД «CustomerType»
2. В настройках данного компонента добавьте в список поле БД «CustomerType.TypeName»
3. Выберите опцию «Показать все записи из таблицы»

Далее настроим кнопки создания, редактирования и удаления записей:

Новый тип клиента

Действие = Новая запись
Форма = frmCustTypeAdd

Редактировать

Действие = Показать запись
Компонент таблицы = TableGrid1
Форма = frmCustTypeAdd

Удалить

Действие = Удалить запись
Компонент таблицы = TableGrid1

Настройка формы **frmCustTypeAdd**:

Edit1(Тип клиента)

TableName = CustomerType
FieldName = TypeName

Button1 (Сохранить)

Действие = Сохранить запись
Компонент для сохранения = Edit1
Таблица БД для сохранения = CustomerType

Настройка формы **Form1**:

Так будет выглядеть форма с новыми компонентами (рис. 63)

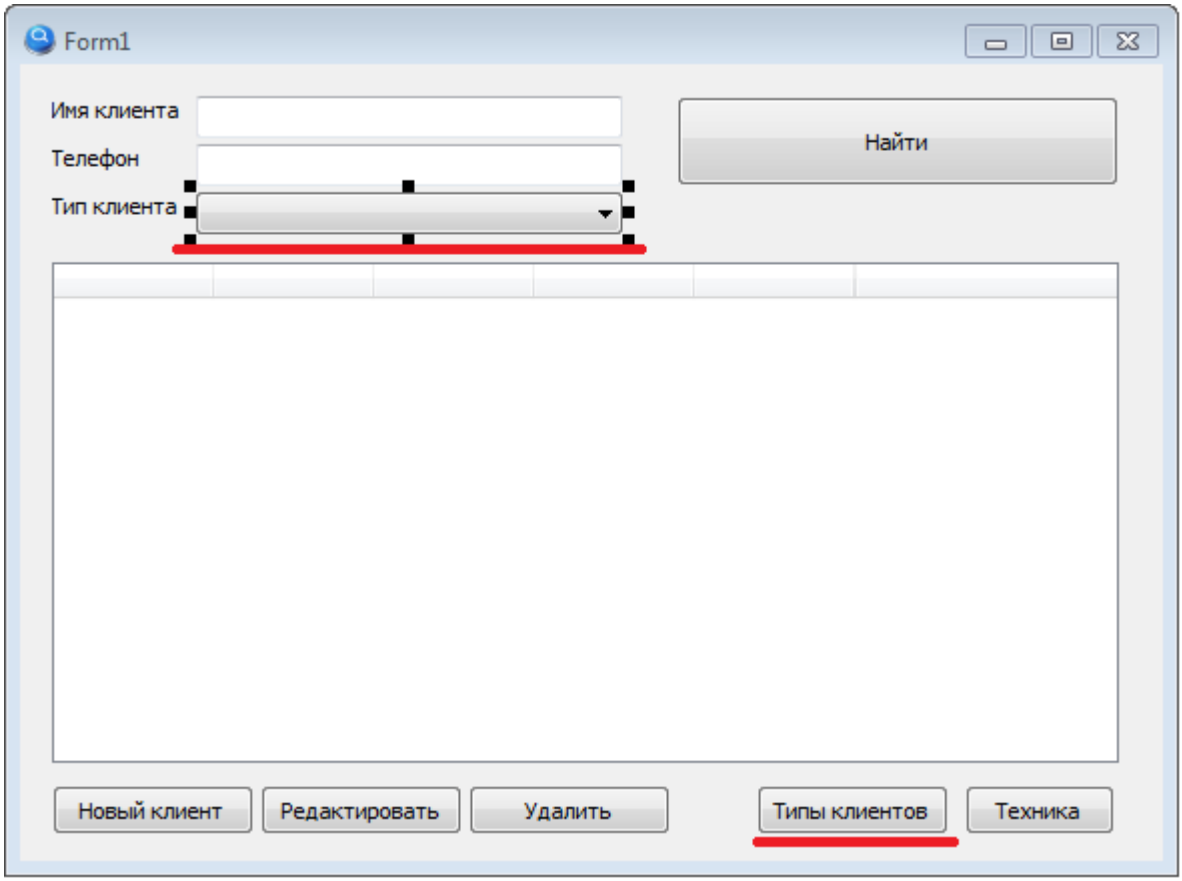


Рис.63

На главную форму добавим компонент ComboBox, чтобы мы смогли фильтровать записи по типу клиентов.

ComboBox1(Тип клиента)
ForeignKey = Client.id_CustomerType
FieldName = TypeName

Далее зайдите в настройки кнопки «Найти», чтобы добавить данный компонент (ComboBox1) в список **«1. Выберите компоненты участвующие в поиске»**, таким образом, данный компонент также будет участвовать в поиске.

Также в список **«3. Формирование результата»** добавьте поле из таблицы БД «CustomerType.TypeName», чтобы в результате поиска мы видели, к какому типу принадлежит клиент.

Расположите на форме новую кнопку, которая будет вызывать форму «frmCustTypeList», чтобы мы могли добавить типы клиентов.

Типы клиентов (кнопка)
Действите = Показать форму
Форма = frmCustTypeList

Настройка формы **frmClient**:

Также как и на предыдущей форме, нам необходимо добавить компонент ComboBox, чтобы мы смогли выбрать, какому типу принадлежит клиент

ComboBox1(Тип клиента)
ForeignKey = Client.id_CustomerType
FieldName = TypeName

И последнее, в настройках кнопки «Сохранить», добавьте компонент ComboBox1 в список **«1.Выберите компоненты, участвующие в сохранении записи»**, чтобы данные с этого компонента сохранялись.

В результате форма будет выглядеть так, как показано на рисунке 64.

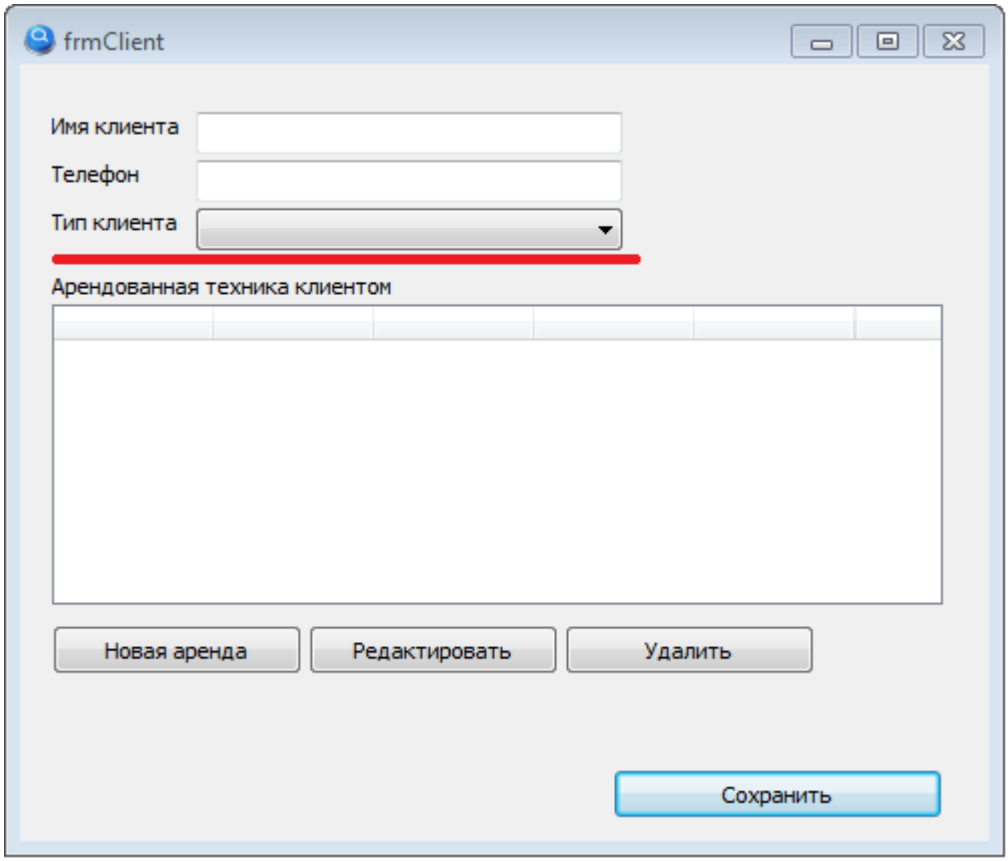


Рис.64

Теперь можете запустить проект и добавить типы клиентов, нажав на главной форме кнопку «Типы клиентов», как показано на рисунке 65.

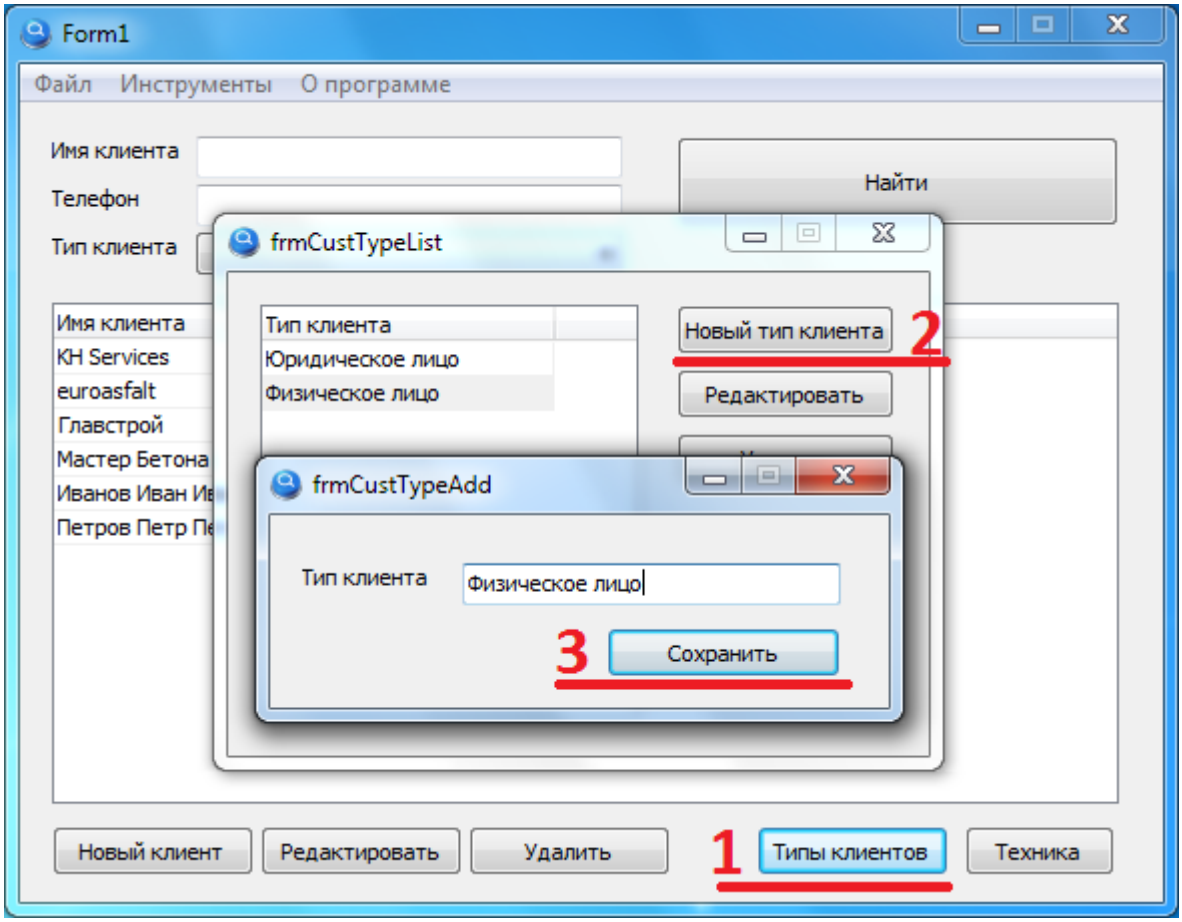


Рис.65

Добавив типы клиентов, можете начать использовать их как для существующих клиентов, так и для новых, указав, какому типу они принадлежат (рис. 66).

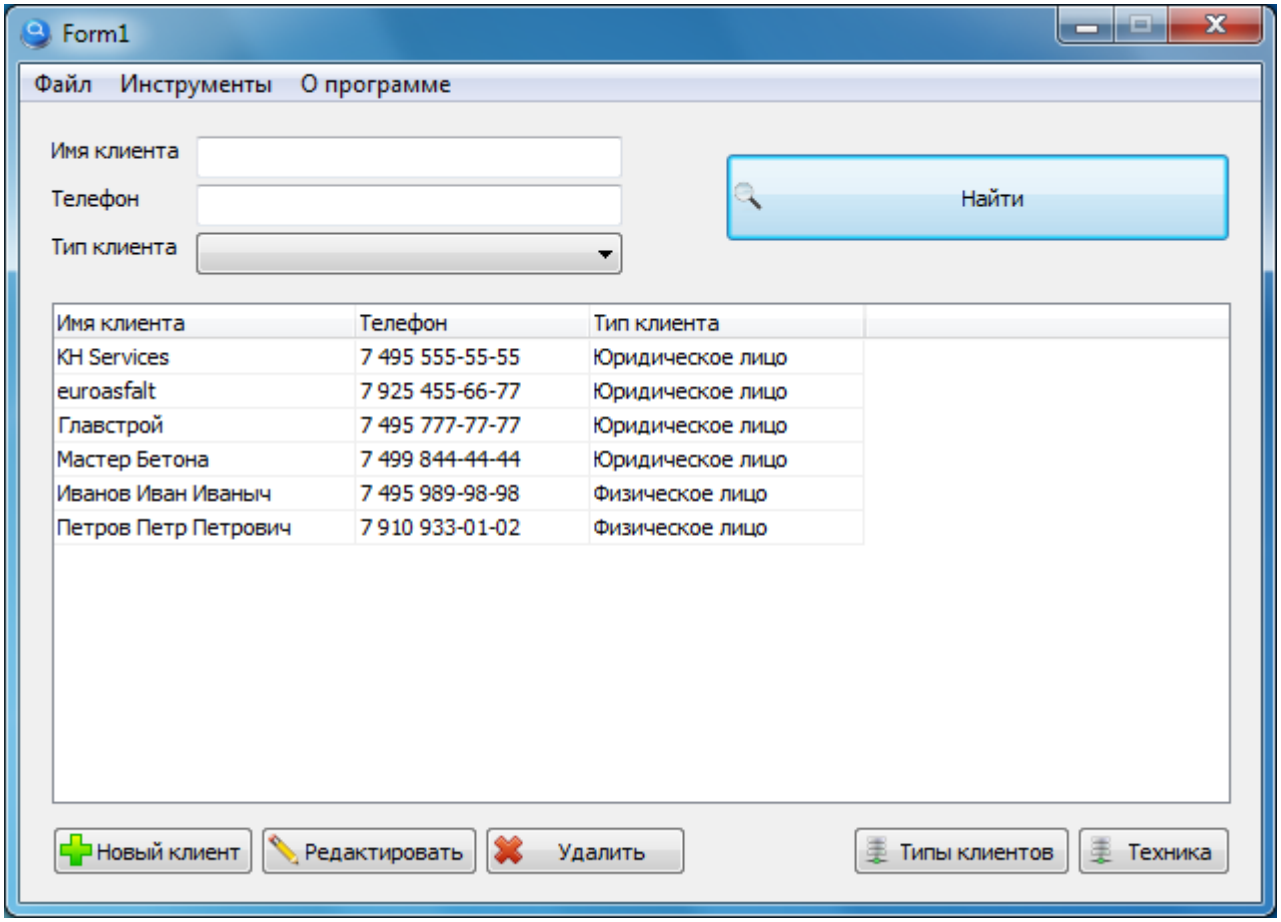


Рис.66

И напоследок, чтобы как то оживить интерфейс, для каждой кнопки можно выбрать свою иконку, используя свойство Icon.

3. Печать.

3.1 Способы печатать.

Рано или поздно вам потребуется распечатать информацию, которая содержится в базе данных. Для этого у нас есть три способа.

1. Самый простой способ печати, это отправить данные содержащие в компоненте TableGrid в программу Microsoft Excel, затем уже встроенными возможностями данной программы распечатать их. Поможет вам в этом кнопка с действием «Открыть в Excel» (рис. 67).

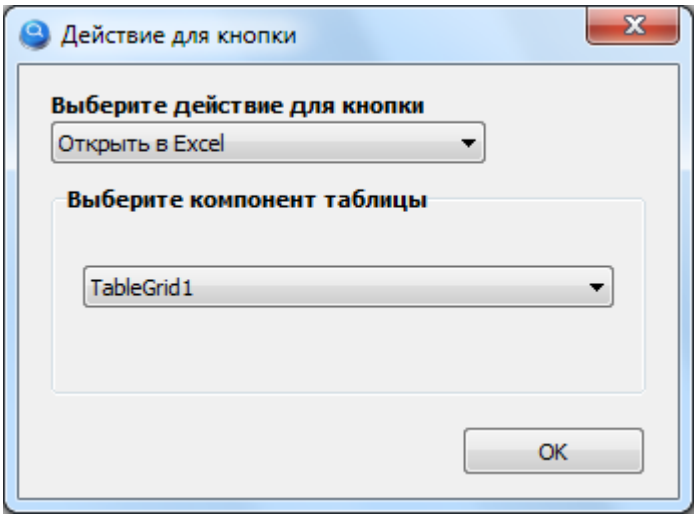


Рис.67

2. Использовать в качестве шаблона печати Microsoft Word документ. Оформив документ определенным образом, можно вставить в него данные в заранее подготовленные места. Данный способ подразумевает использование простого скрипта, поэтому будет рассмотрен в главе, посвященный программированию скриптов.

3. Использовать встроенный дизайнер отчета. Данный способ, наиболее подвинутый и позволяет реализовать практический любой отчет с дальнейшей его печатью или экспортом в другие популярные форматы (doc, xls, pdf и многие другие). Именно этот способ мы подробно рассмотрим в следующей главе.

3.2 Дизайнер отчета

Как уже выше было упомянуто, встроенный дизайнер отчета позволяет реализовать отчет практически любой сложности. Но мы пока рассмотрим только основные его возможности, которых, как правило, достаточно для большинства задач.

В качестве примера реализуем печать простого договора, в котором говорится об ответственности клиента за сохранность арендованной им техники. Выглядеть он будет так, как показано на рисунке 68.

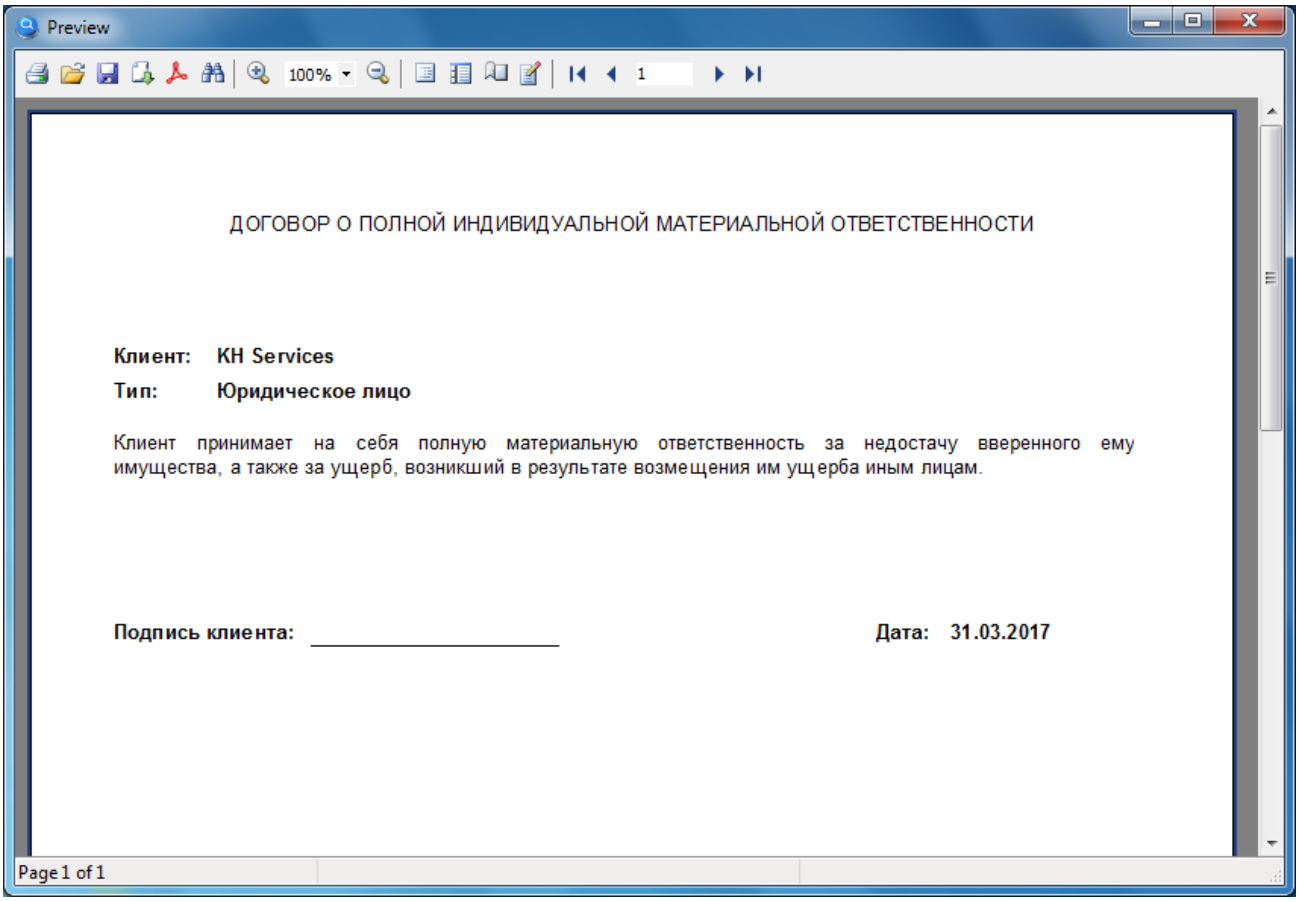


Рис.68

Для создания отчета, с последующей его печатью необходимо использовать кнопку с действием «Отчет». Поместите кнопку на главную форму нашего проекта и назовите ее «Печать ответственности», как показано на рисунке 69. Используя свойство Icon можете выбрать иконку «Printer»

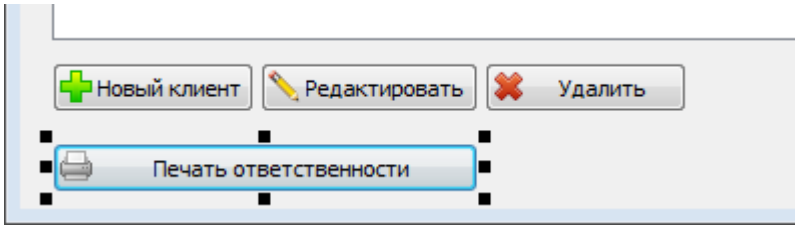


Рис.69

Настройка кнопки с действием «Отчет» практически ничем не отличается от настройки кнопки с действием «Поиск», т.к. по сути оба действия производят поиск по базе данных с последующим выводом результата поиска, только в нашем случае результат поиска мы увидим при печати.

Настройка кнопки «Отчет» показана на рисунке 70.

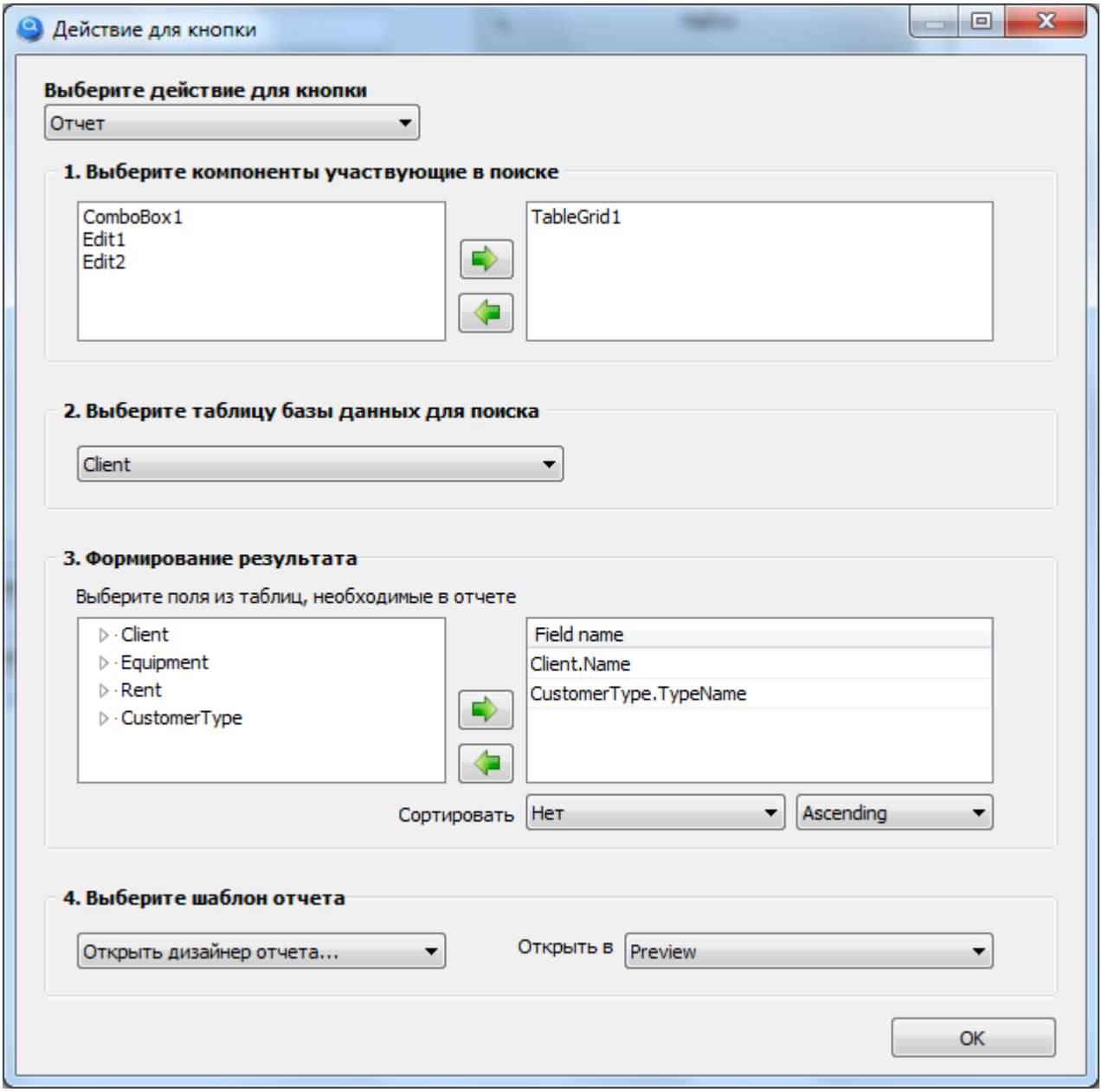


Рис.70

Подробно рассмотрим данную настройку по шагам:

1. Выберите компоненты участвующие в поиске

Здесь выбран компонент TableGrid1, в котором мы видим всех наших клиентов, таким образом, в отчет попадет только тот клиент, который был выделен в компоненте TableGrid1

2. Выберите таблицу базы данных для поиска

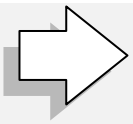
Мы будем печатать информацию о клиенте, соответственно выбираем таблицу БД **Client**.

3. Формирование результата

Выбираем поля таблицы, которые нам необходимы в отчете, в нашем случае это наименование клиента и его тип (поля *Client.Name* и *CustomerType.TypeName* соотвественно).

4. Выберите шаблон отчета

Так как шаблон отчета нам еще предстоит создать, в данной настройке должно быть выбрано «Открыть дизайнер отчета…»



Шаблон отчета - это файл, который определяет, как именно будет выглядеть отчет, данный файл создается в дизайнере отчета и будет сохранен в папке Report вашего проекта.

Чтобы создать шаблон отчета, необходимо запустить наш проект  .

В запущенном проекте нажмите на созданную ранее кнопку «Печать ответственности», таким образом, вы откроете дизайнер отчета.

В создании простого шаблона нет ничего сложного, на рисунке 71 можете видеть, откуда и куда были помещены объекты. Проделайте то же самое.

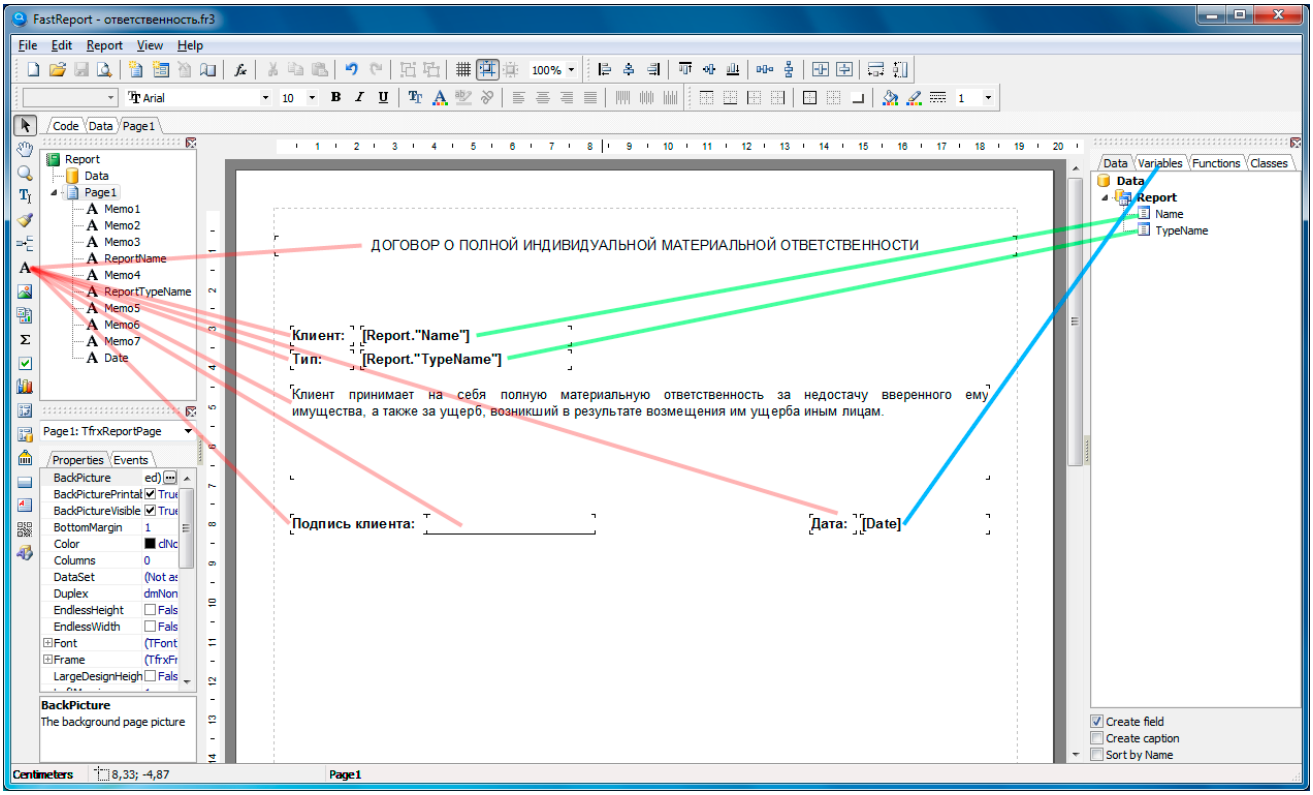


Рис.71

Если все готово, зайдите в меню File > Preview (Ctrl+P) и вы должны увидеть готовый к печати отчет, как было ранее показано на рисунке 68.

Осталось сохранить файл отчета, меню File > Save(Ctrl+S). Назовите файл отчета, например «ответственность», как показано на рисунке 72.

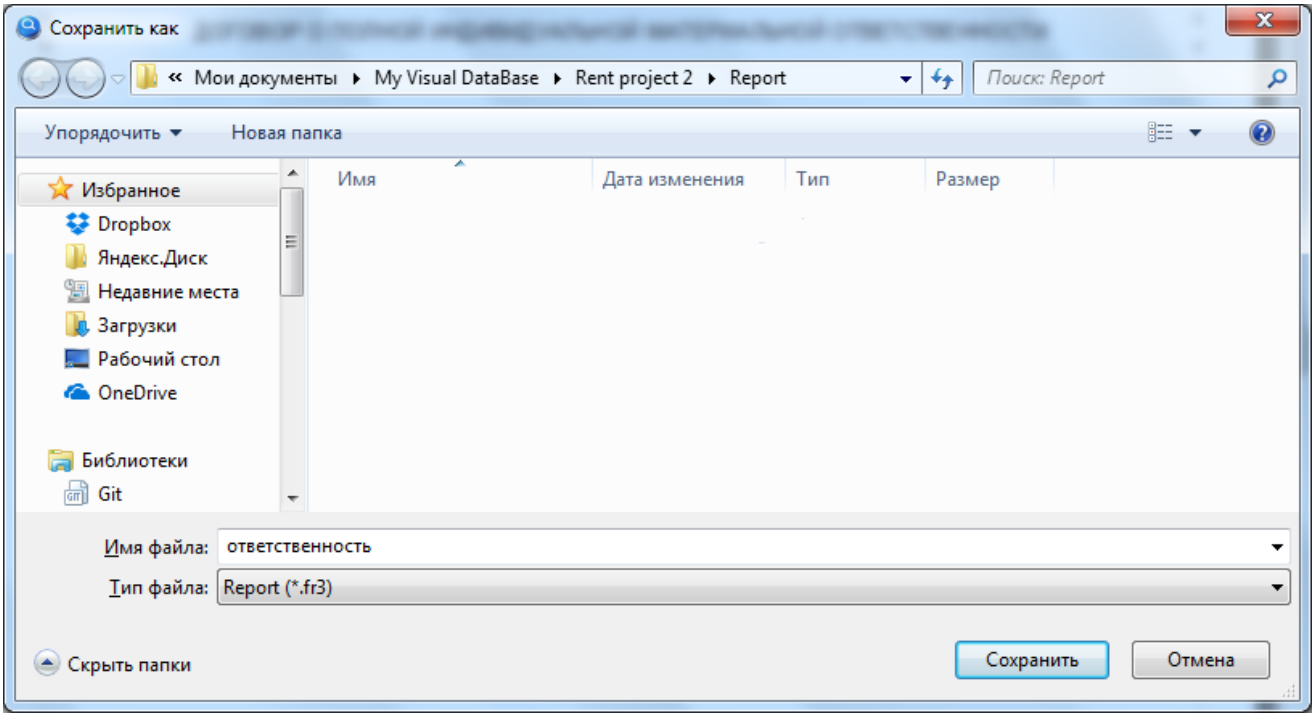


Рис.72

Последний шаг. Закройте запущенный проект и снова зайдите в настройки кнопки «Печать ответственности».

В пункте настроек «**4. Выберите шаблон отчета»**, вместо «Открыть дизайнер отчета…» выберите из списка ранее сохраненный шаблон отчета «ответственность.fr3», как показано на рисунке 73.

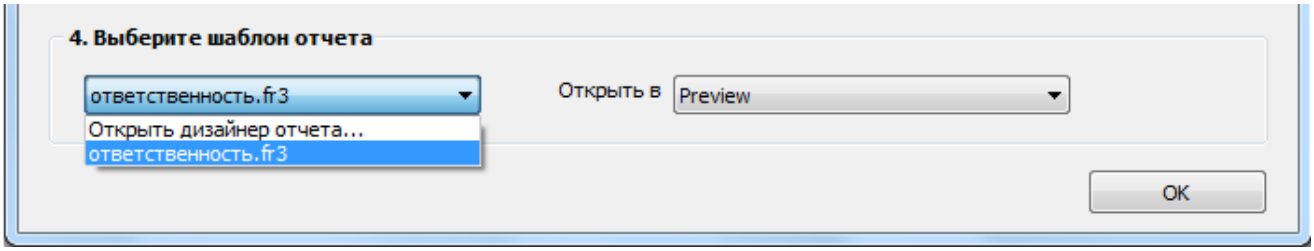


Рис.73

Теперь снова можете запустить проект. Выделив необходимого клиента в компоненте TableGrid1, нажмите кнопку «Печать ответственности», после чего вы увидите готовый к печати отчет с данными выделенного клиента.

Порой бывает удобно, поместить кнопку печати непосредственно на форму добавления/редактирования. Поэтому давайте скопируем кнопку «Печать ответственности» на форму **frmClient**.

В настройках данной кнопки удалите компонент TableGrid1 из списка «**1. Выберите компоненты участвующие в поиске**», т.к. если кнопка с действием «Отчет» размещена на форме предназначенная для добавления/редактирования, она автоматически распознает, какую именно запись необходимо отправить на печать.

Сделаем еще один отчет, но чуть сложней. Теперь нам необходимо распечатать данные о клиенте, со всеми его арендами техники. В результате мы получим отчет, как показано на рисунке 74.

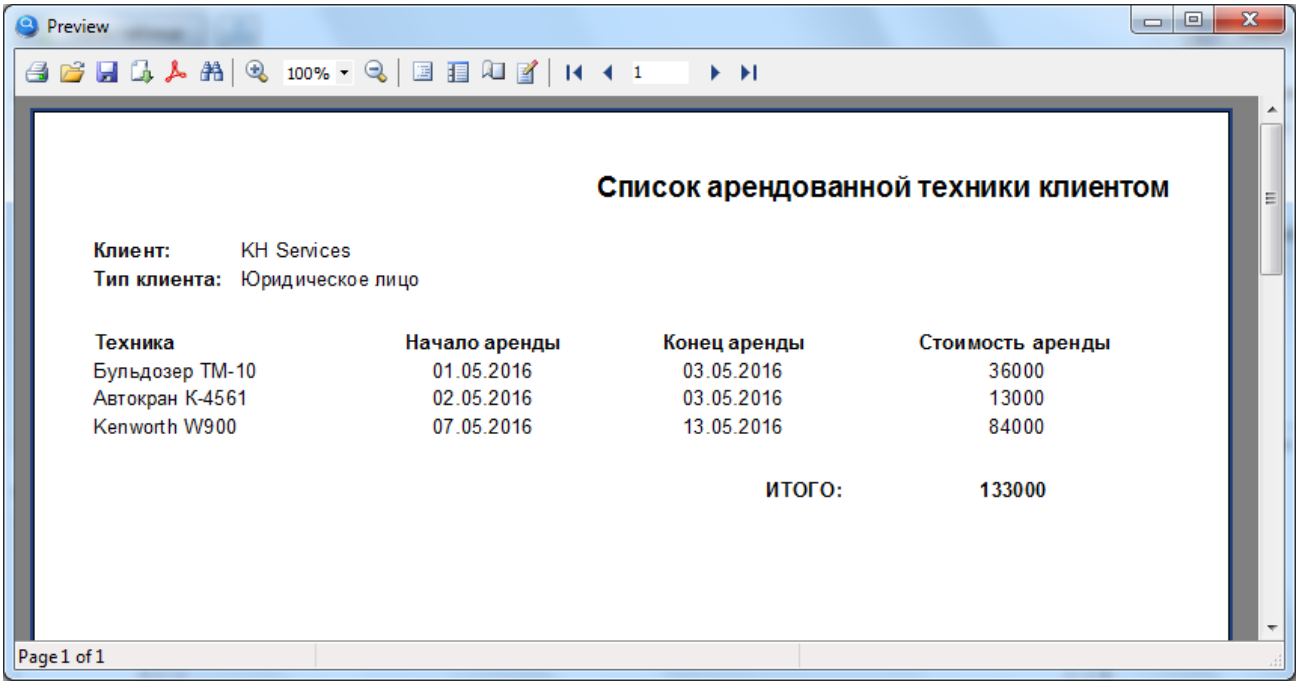


Рис.74

Поместите на главную форму новую кнопку с названием «Арендованная техника», как показано на рисунке 75.

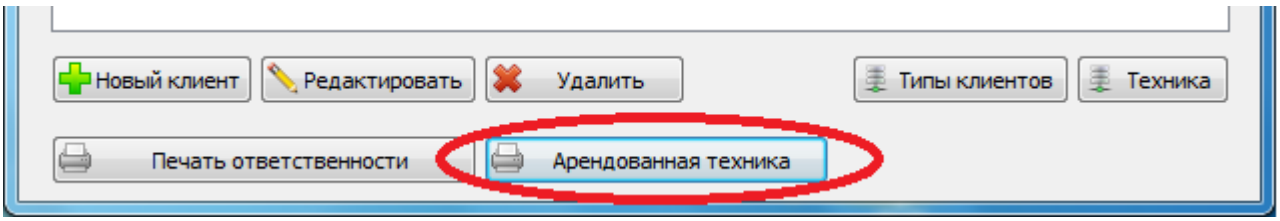


Рис.75

Выберите для данной кнопки действие «Отчет», настройка данной кнопки показана на рисунке 76.

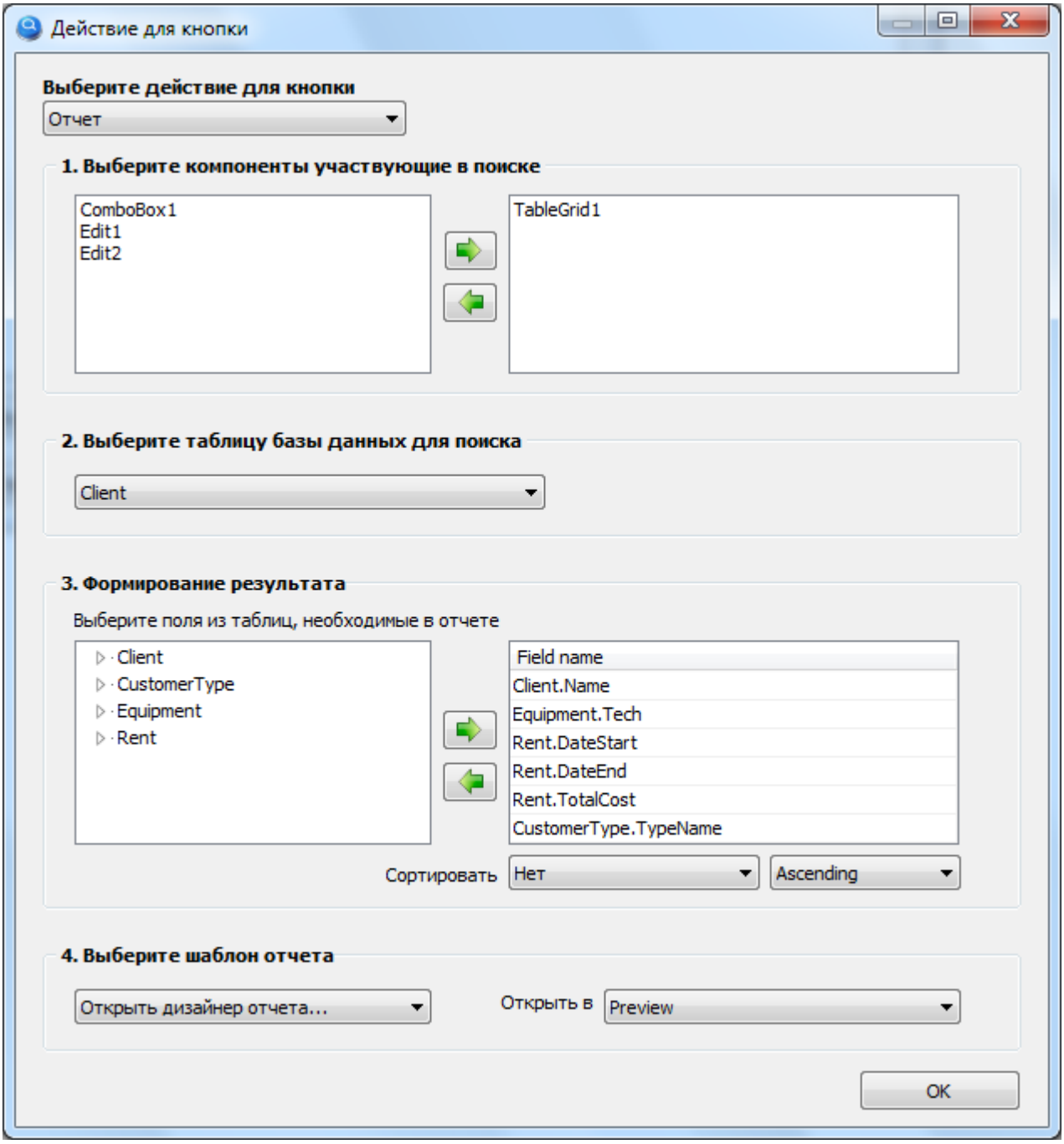


Рис.76

Настройка данной кнопки практически ничем не отличается от настройки кнопки «Печать ответственности», добавлены лишь новые поля в п. «3. Формирование результата», которые нам необходимо видеть в отчете.

Тут стоит упомянуть об одном нюансе. Как видно из настроек, для формирование результата мы выбираем поля из 4 различных таблиц (Client, Equipment, Rent, CustomerType), как правило, в других программах необходимо составлять SQL запрос, в котором вручную указывается, каким образом одна таблица, связывается с другой таблицей и в какой последовательности.

Программа My Visual Database старается самостоятельно понять, как именно необходимо связать таблицы между собой, чтобы получились именно те данные, на которые вы рассчитываете, таким образом, избавляя вас от необходимости изучения языка запросов SQL.

К сожалению не всегда программа может предвидеть, каким образом необходимо связать таблицы, чтобы в результате получить данные, которые вы хотели бы видеть. Такая ситуация может возникнуть, когда вам необходимо связать 3 и более таблицы и если между ними нет очевидных связей. Так, например, между таблицей «Client» и «Equipment» нет связей, связь между этими таблицами осуществляется посредством другой таблицы «Rent».

Что же делать? На вкладке «Таблицы базы данных», под каждой таблицей есть галочка «Таблица является словарем». На данной вкладке необходимо отметить, какие таблицы в вашем проекте являются словарями. Но как понять, какие именно таблицы являются словарями?

Примерами словарных таблиц может служить таблица, в которой содержаться **названия стран**, **статусы** (Открыт, Закрыт), **типы** (Юридическое лицо, Физическое лицо), **наименование с ценами**, и т.д.

То есть, такие таблицы, которые, как правило, заполняются первыми при начале работы с БД и в дальнейшем не редактируются либо редактируются редко.

В нашем проекте к словарными таблицам можно отнести «CustomerType» и «Equipment», установите галочку «Таблица является словарем» под данными таблицами, таким образом, вы поможете программе правильно связать таблицы.

И так, продолжаем создавать отчет об арендованной техники клиентом.

Как вы помните, чтобы создать шаблон отчета, необходимо запустить наш проект  .

В запущенном проекте нажмите на созданную ранее кнопку «Арендованная техника», таким образом, вы откроете дизайнер отчета.

Данный отчет будет чуть сложней, чем предыдущий, так как в нем будут использованы так называемые блоки (или Band). Блоки позволяют создать почти любую структуру отчета, с некоторыми из них мы сейчас познакомимся.

Чтобы посмотреть все доступные блоки, нажмите слева иконку  , после чего вы увидите меню, как показано на рисунке 77.

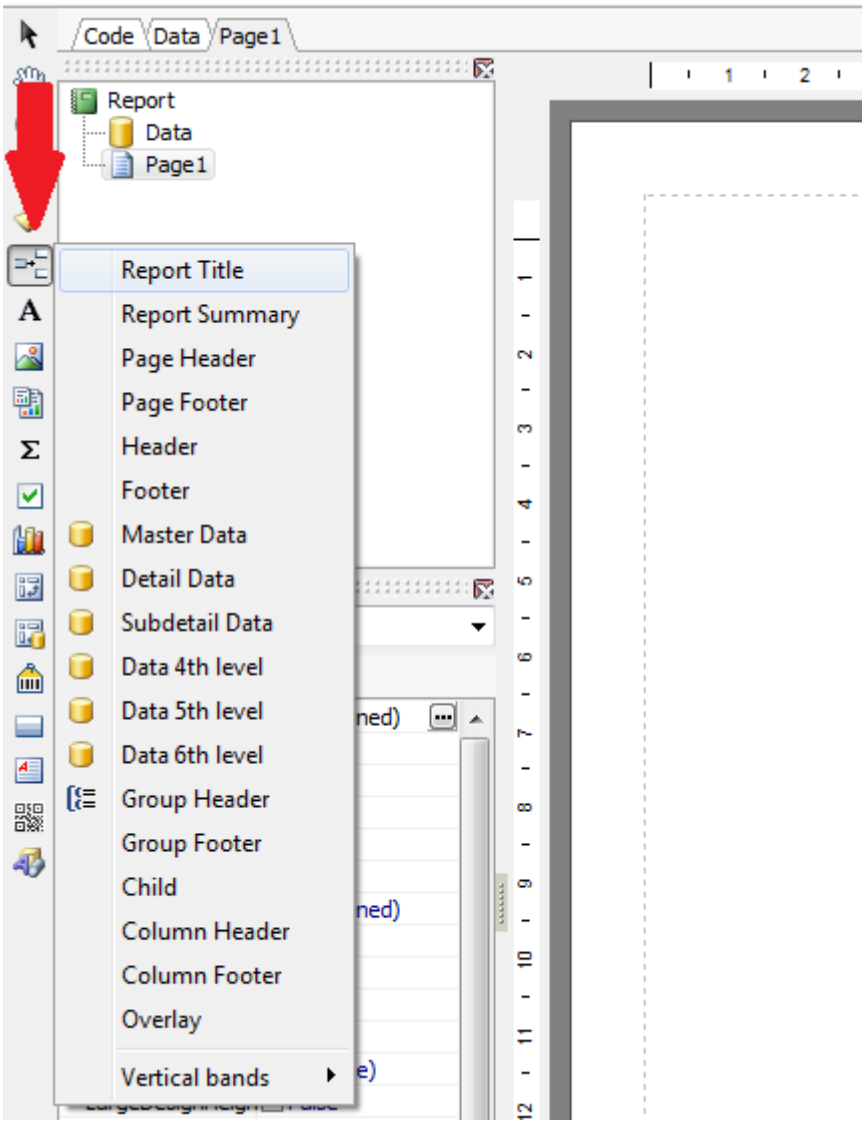


Рис.77

Приступаем к созданию отчета с помощью блоков, попутно узнаем, для чего они.

Из меню, показанного на рисунке 77, выберите блок **Report Title**, после чего данный блок появится в отчете. В данном блоке вы можете поместить заголовок отчета, например его название. Информация, размещенная в данном блоке, будет распечатана только на первой странице вашего отчета.

Поместим в данный блоке название отчета. В результате у вас должно получиться, как показано на рисунке 78.

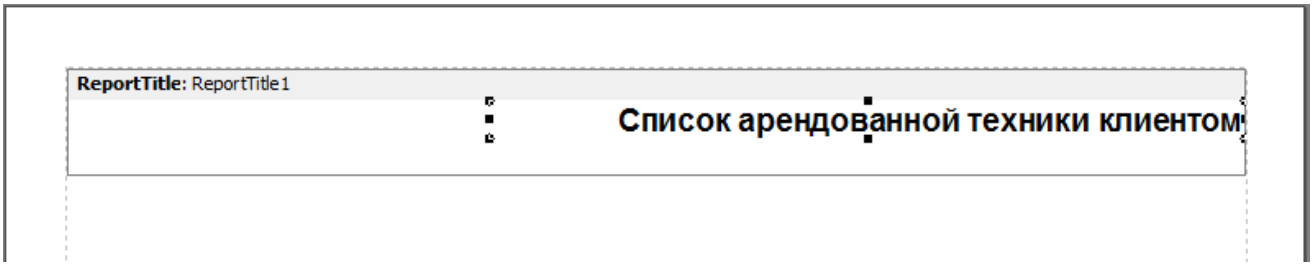


Рис.78

Поместите следующий блок **PageHeader**. Информация, размещенная в данном блоке, будет выведена на каждой распечатанной странице (в случае если ваш отчет не помешается на одной странице).

В данном блоке мы поместим информацию о клиенте. Также поместим заголовки для таблицы, которая будет расположена в следующем блоке.

Поместите в данный блок текст и поля с данными, как показано на рисунке 79.

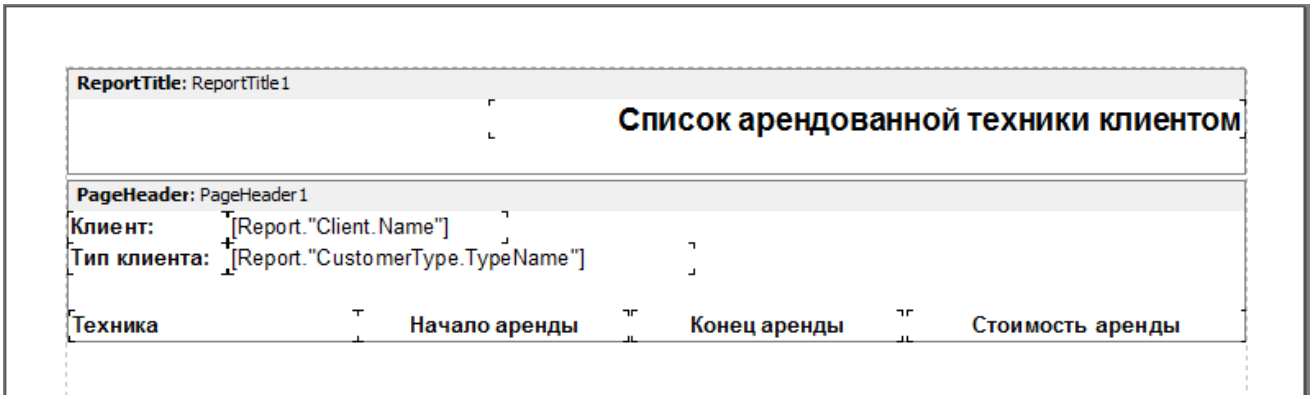


Рис.79

Следующий блок, который необходимо поместить в отчет, это **Master Data**.

Данный блок предназначен для вывода информации в виде таблицы. С помощью данного блока мы получим таблицу, в которой будет перечислена вся техника, которая была арендована клиентом.

Поместите данный блок в отчет, выбрав его из меню. Перед тем как он появится в отчете, вы увидите окно с заголовком **Select DataSet**, в котором необходимо выбрать источник данных. Выберите из списка источник данных с названием **Report** и нажмите **OK**.

В данном блоке необходимо поместить поля базы данных, из которых будет сформирована таблица. В результате у вас должно получиться, как показано на рисунке 80.

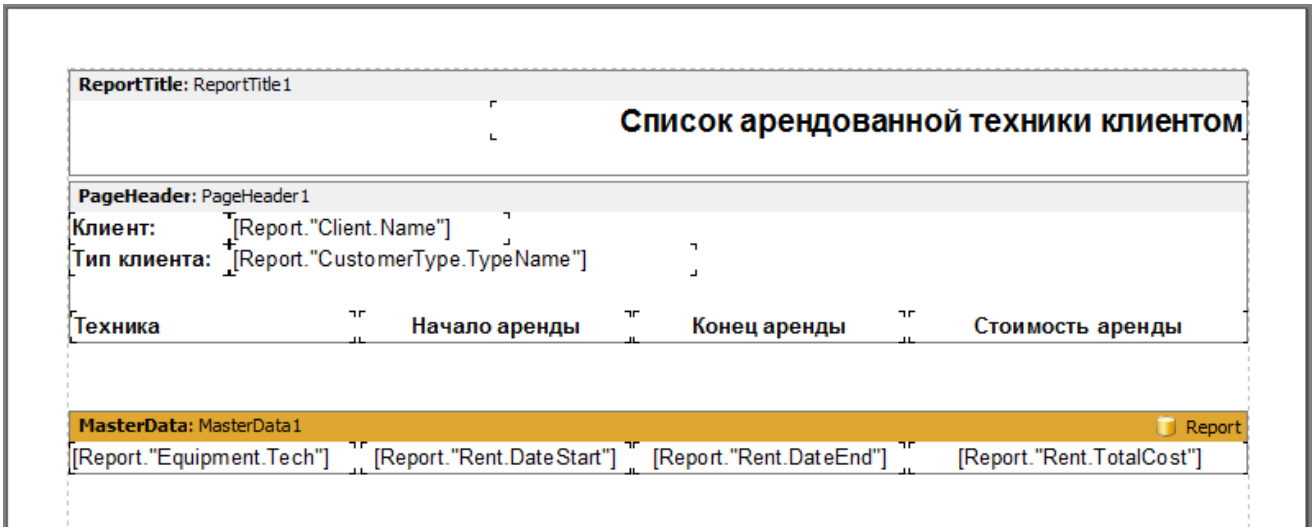


Рис.80

Книга все еще находится в процессе написания.

Последнюю версию книгу вы можете найти здесь http://myvisualdatabase.com/book_ru.html

Также по данной ссылке вы можете написать отзыв о книге, либо свой задать вопрос.

Спасибо.