

ЛАБОРАТОРНАЯ РАБОТА №4 (ЗАДАНИЕ 1)

Технология защиты информации.

ДОБАВЛЕНИЕ НОВОЙ ТАБЛИЦЫ В БАЗУ ДАННЫХ

Для выполнения лабораторной работы №8 нам потребуется создать новую таблицу в базе данных **aero**.

Откройте **SQL Server Management Studio**. В обозревателе объектов найдите базу данных **aero**. Нажмите правой кнопкой мыши на узле «**Таблицы**» и выберите пункт всплывающего меню «**Таблица...**».

Добавьте в новую таблицу следующие столбцы:

- **Id** – целочисленное значение (**int**)
- **Login** – строка символов с максимальной длиной 256 букв (**varchar(256)**)
- **Password** – строка символов с максимальной длиной 256 букв (**varchar(256)**)

В конструкторе таблиц уберите все галочки в столбце «**Разрешить значения NULL**», это сделает столбцы обязательными к заполнению.

Назначьте столбец **Id** первичным ключом с автоматическим приращением (см. лабораторную работу №4, стр. 8-9).

Сохраните таблицу под именем **Users**.

ШИФРОВАНИЕ ПАРОЛЕЙ

Как вы уже поняли, таблица **Users** предназначена для хранения паролей пользователей. Однако мы не будем хранить пароли в явном виде, в целях безопасности все пароли в базе данных будут зашифрованы. Тогда, если злоумышленник получит содержимое таблицы **Users**, то хранящиеся там пароли будут ему по-прежнему не известны.

В нашей программе мы будем осуществлять шифрование при помощи преобразования пароля в MD5 хэш. Шифрование необратимое, получить обратно пароль из хэша невозможно. Функция для преобразования текста в хэш выглядит следующим образом:

```
string GetMD5(string value) {  
    var provider = new System.Security.Cryptography.MD5CryptoServiceProvider();  
    var data = Encoding.UTF8.GetBytes(value);  
    data = provider.ComputeHash(data);  
    return Encoding.UTF8.GetString(data);  
}
```

Метод принимает в качестве единственного аргумента произвольную строку символов.

Внутри создается объект **provider** класса **MD5CryptoServiceProvider**. Этот класс занимается вычислением хэша при помощи системного модуля криптографии.

```
var provider = new System.Security.Cryptography.MD5CryptoServiceProvider();
```

Для вычисления используется метод **ComputeHash**, принимающий в качестве аргумента массив байт (**byte[]**). Передаваемая строка **value** состоит из *двухбайтовых* символов **Unicode**, поэтому перед шифрованием нам необходимо ее преобразовать. Для этого воспользуемся классом **Encoding**, и преобразуем строку **value** в однобайтовое представление в формате **UTF-8**.

```
var data = Encoding.UTF8.GetBytes(value);
```

В принципе, мы можем использовать любое другое преобразование текста в массив байт, например преобразование из **Unicode** в **ASCII**. Для нас это не имеет значения, поскольку смысловое содержание строки **value** пропадет после шифрования. Главное, чтобы полученный массив байт был уникален.

Шифруем массив байт **data**:

```
data = provider.ComputeHash(data);
```

Для удобства отправки в базу данных преобразуем массив байт обратно в строку и вернем полученное значение:

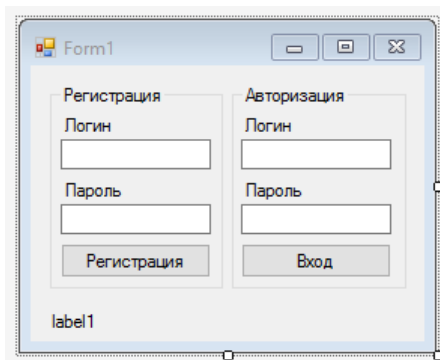
```
return Encoding.UTF8.GetString(data);
```

ПРИЛОЖЕНИЕ WINDOWS FORMS

Шифрование паролей широко применяется при использовании архитектуры клиент-сервер. Для удобства, в данной лабораторной работе мы реализуем процедуры регистрации и аутентификации в одном приложении.

Создайте новый проект **Windows Forms** и назовите его Lab8.

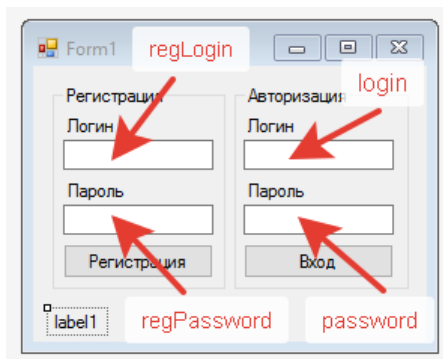
В редакторе форм разместите элементы управления по следующей схеме:



Для выделения в окне отдельной области используется элемент **GroupBox**. Особой функциональной нагрузки эти элементы не несут, их роль исключительно декоративная.

Для создания подписей используется элемент **Label** (метка). Разместите подписи по своему усмотрению и добавьте одну метку в левый-нижний угол формы. В свойствах у этой метки укажите имя объекта (**Name**) как **debugLabel**. Мы будем использовать ее в отладочных целях для отображения результатов шифрования.

За поля ввода текста отвечает элемент управления **TextBox**. Создайте на форме 4 таких элемента и в их свойствах присвойте им следующие имена: **regLogin**, **regPassword**, **login**, **password**.



Для кнопок «Регистрация» и «Вход» поменяйте имена переменных на **Create** и **Enter** соответственно.

Дважды щелкните на кнопку «Регистрация» для создания обработчика события нажатия на кнопку.

Скопируйте код метода **GetMD5** из параграфа «Шифрование паролей» и вставьте в код формы.

Создайте новое подключение LINQ to SQL к базе данных **aero** (см. лабораторную работу №3). В конструктор таблиц достаточно перетащить только таблицу **Users**.

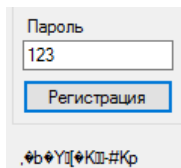
Вернемся к коду формы. Если вы все сделали правильно, то теперь у нас в классе формы **Form1** есть метод **Create_Click** обрабатывающий нажатие на кнопку «Регистрация». Для начала возьмем значение из поля пароль и зашифруем его:

```
var hash = GetMD5(regPassword.Text);
```

Отобразим результат в тексте отладочной метки **debugLabel**:

```
debugLabel.Text = hash;
```

Если мы сейчас запустим программу, то введя какие-либо данные в поле «Пароль» и нажав на кнопку «Регистрация» мы увидим внизу формы результаты шифрования:



Сохраним имя пользователя и пароль в базу данных. Создаем новое подключение:

```
var db = new AeroDataContext();
```

Создадим нового пользователя:

```
var data = new User {  
    Login = regLogin.Text,  
    Password = hash  
};
```

Сохраним пользователя в базе данных:

```
db.Users.InsertOnSubmit(data);  
db.SubmitChanges();
```

Теперь, если мы введем в качестве имени пользователя «**first**», а в качестве пароля «**123**», то таблица **Users** в базе данных станет выглядеть следующим образом:

Результаты		Сообщения	
Id	Login	Password	
1	first	.?b?Y?K?#Kp	

Реализуем процедуру проверки введенного пароля. Перейдите в конструктор форм и дважды щелкните на кнопке «Вход». Метод-обработчик события будет называться **Enter_Click**. Создадим новое подключение:

```
var db = new AeroDataContext();
```

Зашифруем вводимый пароль:

```
var hash = GetMD5(password.Text);
```

Попробуем найти в таблице **Users** строку с подходящими значениями:

```
var query = from c in db.Users  
            where (c.Login == login.Text && c.Password == hash)  
            select c;
```

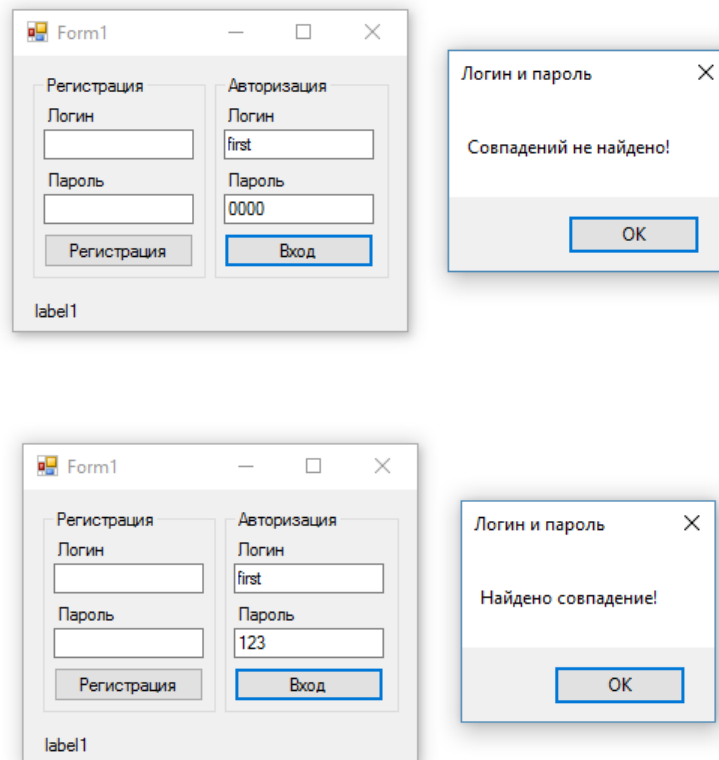
Если в результате выполнения запроса мы нашли одну подходящую строку, то выводим сообщение об успехе, иначе выводим сообщение о неудаче:

```

if (query.Count() == 1) {
    MessageBox.Show("Найдено совпадение!", "Логин и пароль");
} else {
    MessageBox.Show("Совпадений не найдено!", "Логин и пароль");
}

```

Результат работы программы:



ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

В интернете существуют большие хакерские базы зашифрованных паролей. При наличии вычислительных ресурсов, злоумышленник может методом перебора попытаться определить настоящий пароль по украденному хэшу.

- Добавьте в процедуру регистрации проверку на наличие в пароле одновременно строчных букв, заглавных букв и цифр.
- Добавьте в программу «защиту от дурака». При регистрации данные не должны отправляться в базу, если поля «Логин» или «Пароль» не заполнены.
- Добавьте в программу защиту от регистрации нескольких одинаковых пользователей с разными паролями.

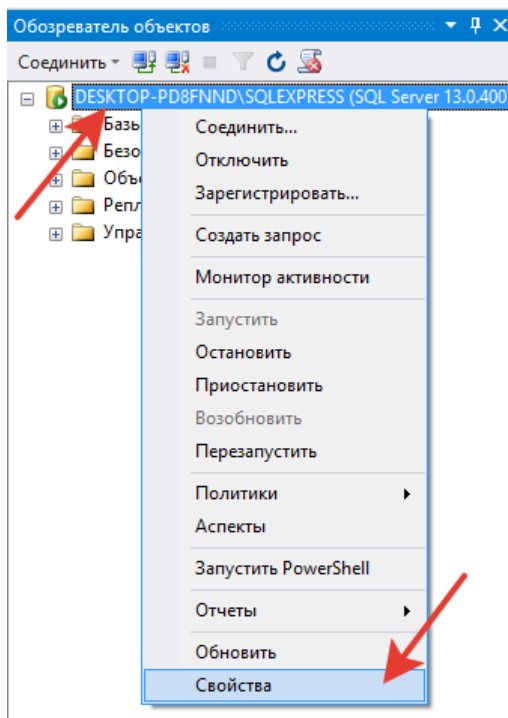
ЛАБОРАТОРНАЯ РАБОТА №4 (ЗАДАНИЕ 2)

Технология защиты информации.

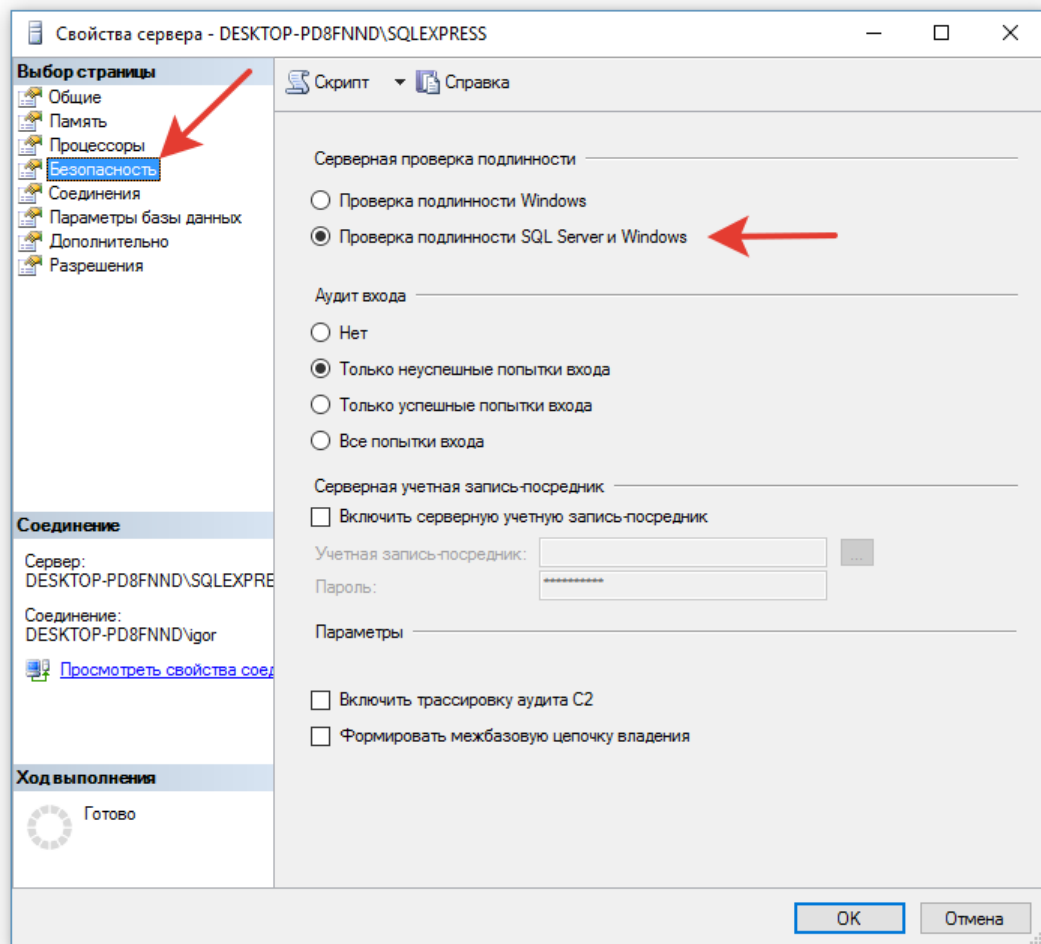
ДОБАВЛЕНИЕ НОВОГО ПОЛЬЗОВАТЕЛЯ В БАЗУ ДАННЫХ

Запустите **SQL Server Management Studio**.

В **обозревателе объектов** нажмите правой кнопкой мыши на узле сервера и выберите пункт выпадающего меню «**Свойства**».

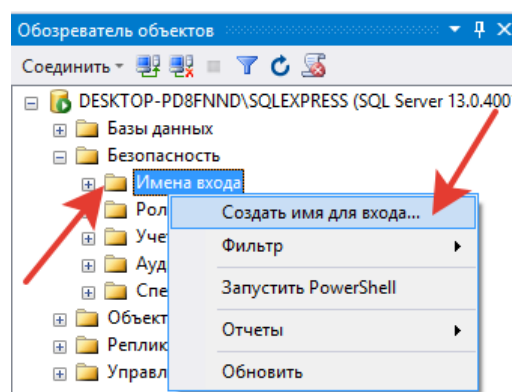


Откройте вкладку «**Безопасность**». На открывшейся странице, в разделе «**Серверная проверка подлинности**» переключите выбор на пункт «**Проверка подлинности SQL Server и Windows**». Нажмите кнопку «**ОК**».



Откроется окно с предупреждением о необходимости перезагрузки сервера, еще раз нажмите кнопку «**ОК**». Затем, в **обозревателе объектов** еще раз нажмите правой кнопкой мыши на **узле сервера** и выберите пункт выпадающего меню «**Перезапустить**».

Дождитесь перезапуска сервера и в **обозревателе объектов** раскройте вкладку «**Безопасность**» (**Security**), нажмите правой кнопкой мыши по вкладке «**Имена входа**» (**Logins**) и в контекстном меню выберите «**Создать имя входа...**» (**New Login...**):



Откроется окно **создания имени входа (Login — New)**. Теперь необходимо определиться с вариантом аутентификации нового пользователя. Возможны 2 варианта:

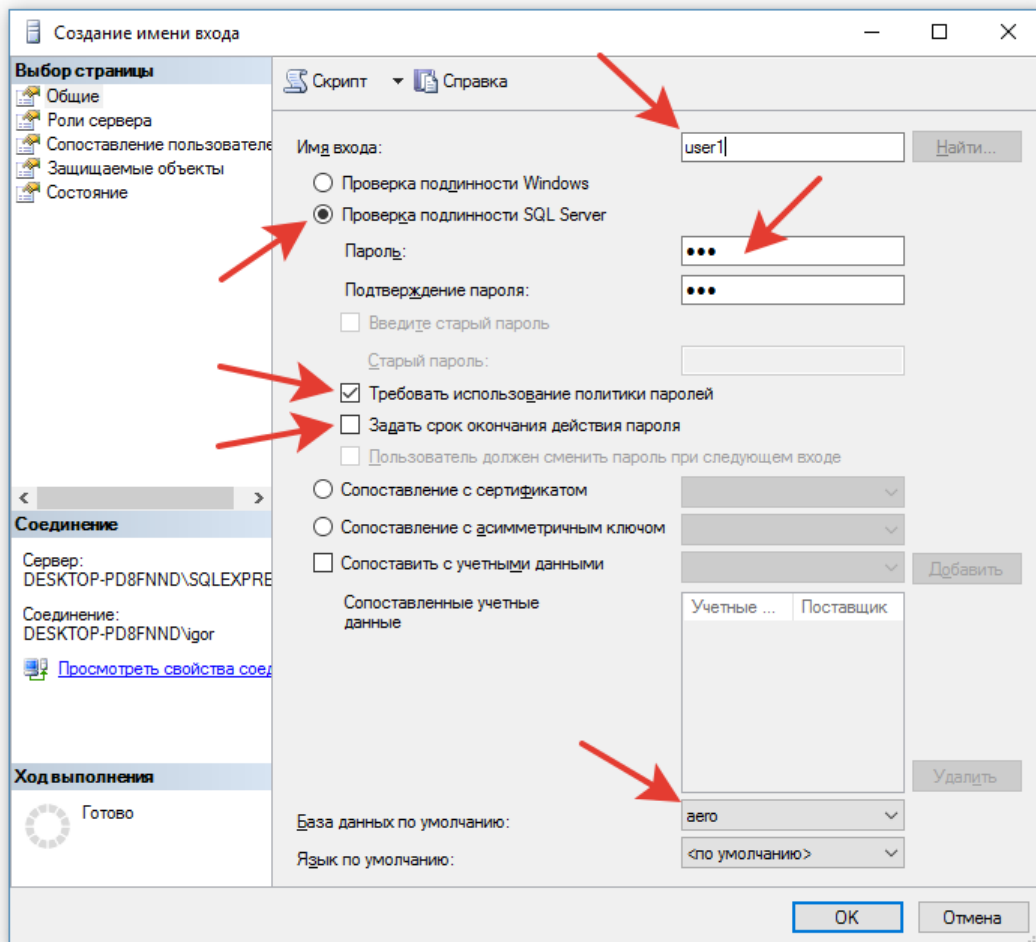
- Аутентификация с помощью пароля — Проверка подлинности SQL Server (SQL Server Authentication).
- Доступ для конкретного пользователя Windows — Проверка подлинности Windows (Windows authentication).

Выберите первый вариант – проверка подлинности SQL Server .

Укажите **имя входа (Login name)**, пусть это будет **user1**. Введите пароль (**Password**) пользователя. Далее:

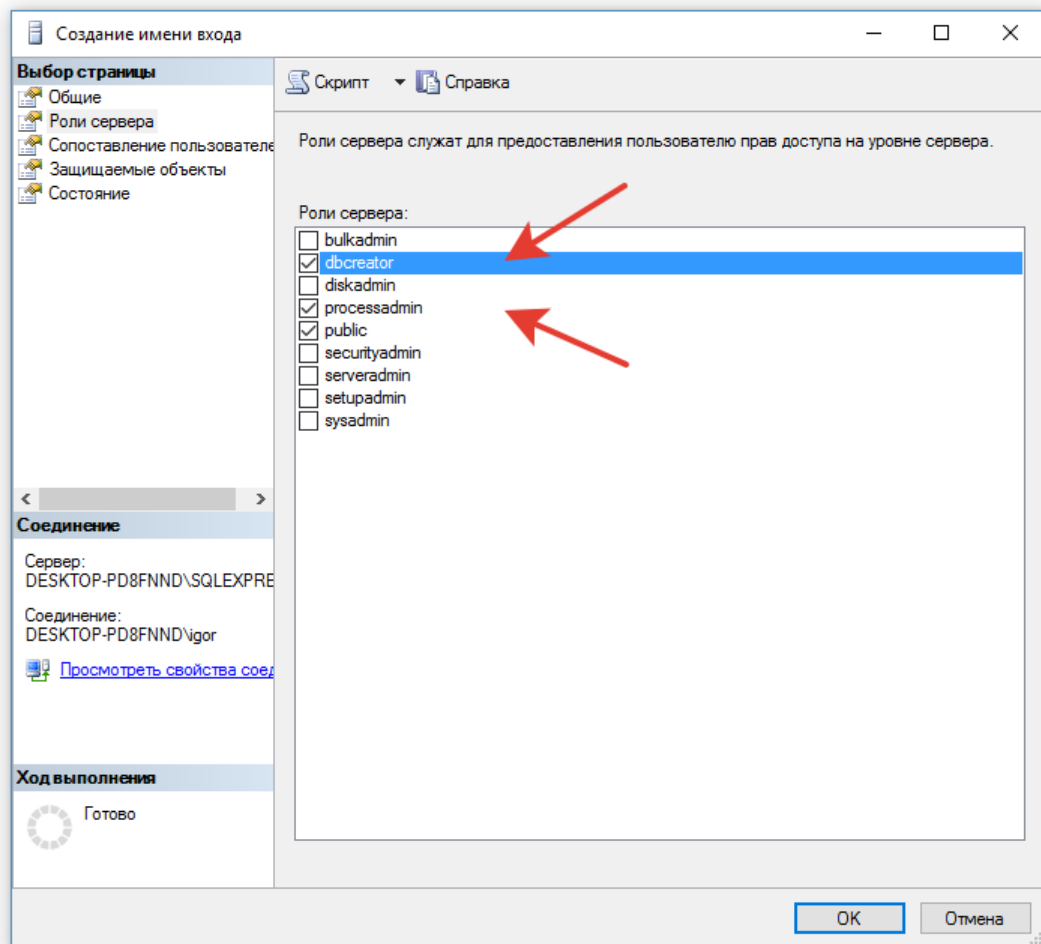
- Поставьте галочку у параметра «**Требовать использование политики паролей**» (**Enforce password policy**);
- Снимите галочку «**Задать срок окончания действия пароля**» (**Enforce password expiration**);
- Снимите галочку «**Пользователь должен сменить пароль при следующем входе**» (**User must change password at next login**)

В пункте «**База данных по умолчанию**» выберите пункт выпадающего списка «**aero**».



В этом же окне перейдите на вкладку «**Роли сервера**» (**Server Roles**).

Здесь необходимо выбрать набор прав добавляемого пользователя. Для текущей задачи отметьте следующие роли: **dbcreator**, **processadmin**, **public**. Нажмите кнопку «**ОК**».



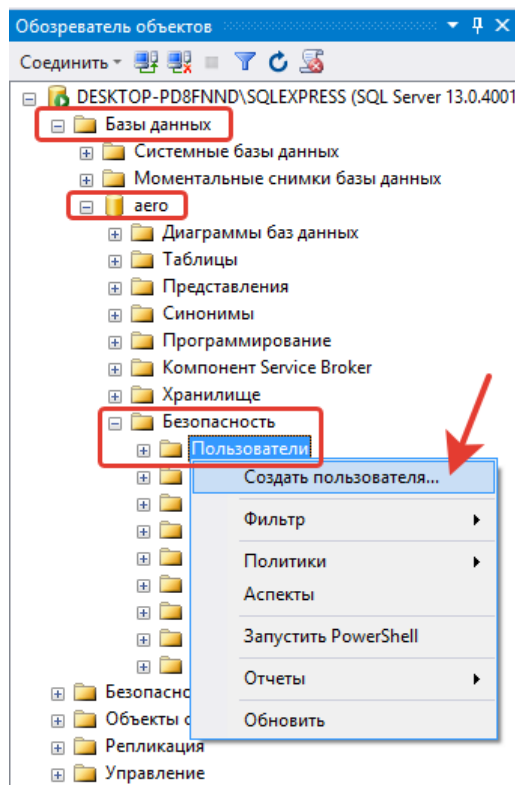
Microsoft SQL Server включает в себя 9 предопределенных серверных **ролей**. Данные роли определены на уровне сервера и поэтому существуют вне баз данных. Они предусмотрены для удобства администрирования MS SQL Server и для обратной совместимости. **Разрешения**, назначенные предопределенным ролям сервера, не могут быть изменены.

В таблице представлены все предопределенные роли уровня сервера и описаны их возможности:

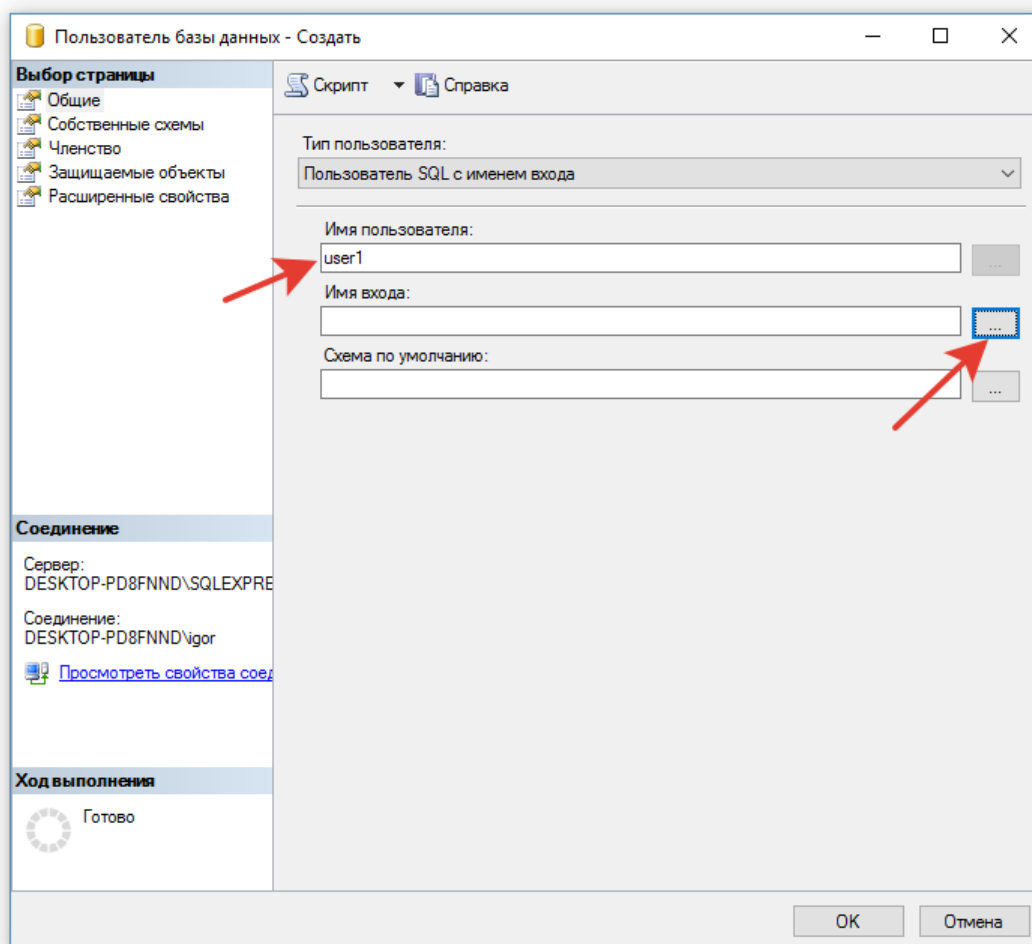
Предопределенная роль уровня сервера	Описание
sysadmin	Члены предопределенной роли сервера sysadmin могут выполнять любые действия на сервере.
serveradmin	Члены предопределенной роли сервера serveradmin могут изменять параметры конфигурации на уровне сервера, а также выключать сервер.
securityadmin	Члены предопределенной роли сервера securityadmin управляют именами входа и их свойствами. Они могут предоставлять, запрещать и отменять разрешения на уровне сервера (инструкции GRANT , DENY и REVOKE). Они также могут предоставлять, запрещать и отменять разрешения на уровне базы данных (инструкции GRANT , DENY и REVOKE) при наличии доступа к базе данных. Кроме того, они могут сбрасывать пароли для имен входа SQL Server.
processadmin	Члены предопределенной роли сервера processadmin могут завершать процессы, выполняемые на экземпляре SQL Server.
setupadmin	Члены предопределенной роли сервера setupadmin могут добавлять или удалять связанные серверы с помощью инструкций Transact-SQL.
bulkadmin	Члены предопределенной роли сервера bulkadmin могут выполнять инструкцию BULK INSERT.
diskadmin	Предопределенная роль сервера diskadmin используется для управления файлами на диске.
dbcreator	Члены предопределенной роли сервера dbcreator могут создавать, изменять, удалять и восстанавливать любые базы данных.
public	Все пользователи, группы и роли SQL Server по умолчанию принадлежат предопределенной роли сервера public .

Создадим **нового пользователя** базы данных **aero** на основе созданного **имени входа**.

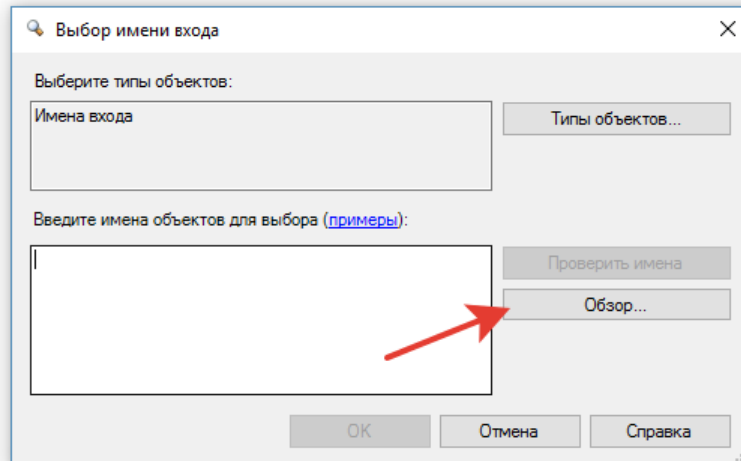
Раскройте узлы «Базы данных» - «аегио» - «Безопасность». Нажмите правой кнопкой мыши на узле «Пользователи» и выберите пункт всплывающего меню «Создать пользователя».



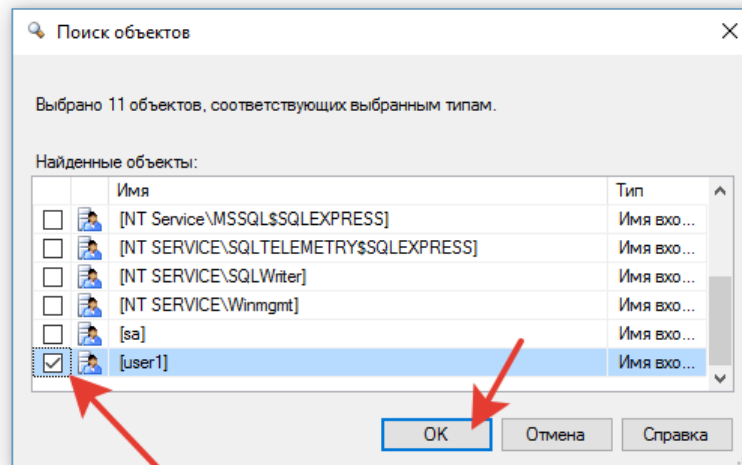
В открывшемся окне введите «user1» в качестве имени пользователя. Затем нажмите на кнопку «...» справа от поля «Имя входа»:



В открывшемся окне нажмите на кнопку «**Обзор**»:

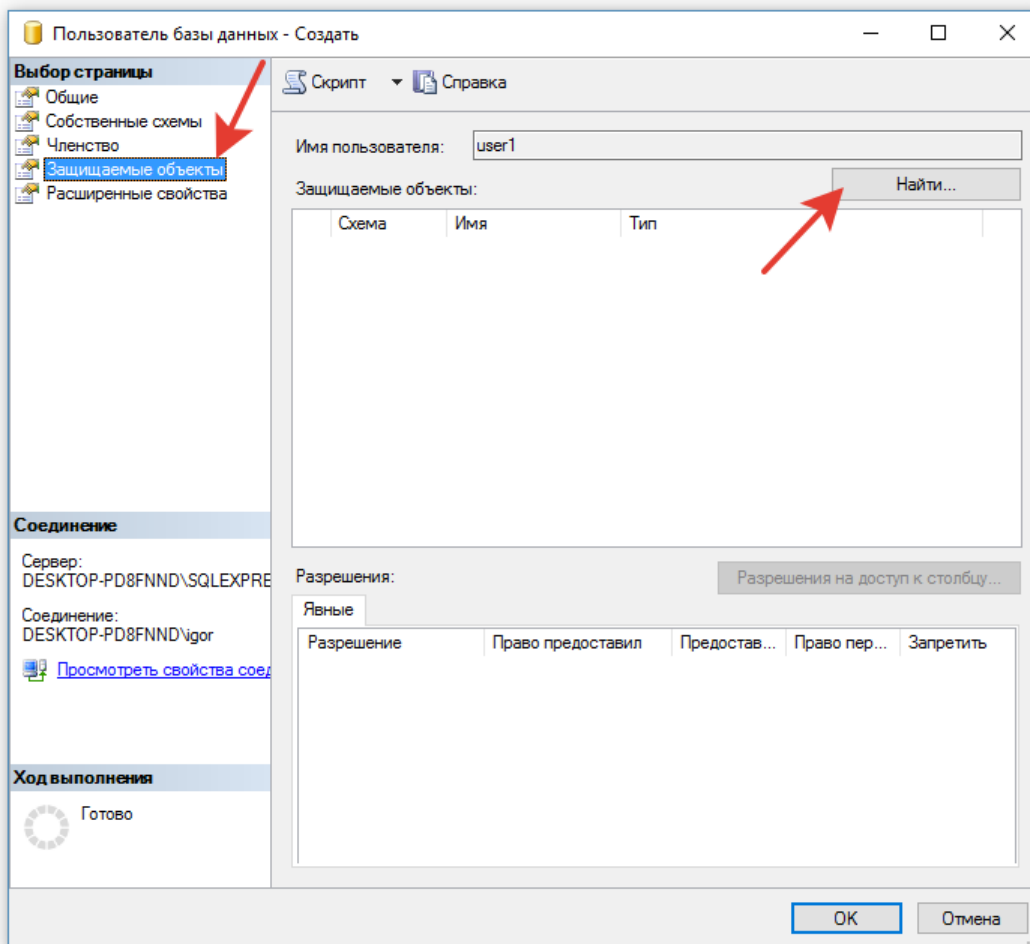


В открывшемся окне найдите в списке строку с именем входа «**user1**», поставьте галочку рядом с именем. Нажмите кнопку «**ОК**»:

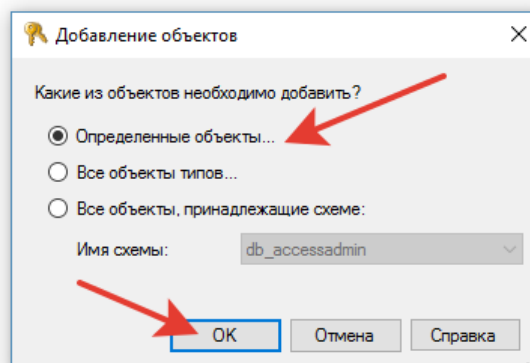


Еще раз нажмите кнопку «**ОК**», чтобы вернуться к окну создания нового пользователя. Теперь наш новый пользователь базы данных связан со своим именем входа, т.е. может аутентифицироваться в базе данных. Однако, после входа в систему он не сможет увидеть содержимое базы, т.к. у него нет соответствующих прав доступа. Назначим их ему.

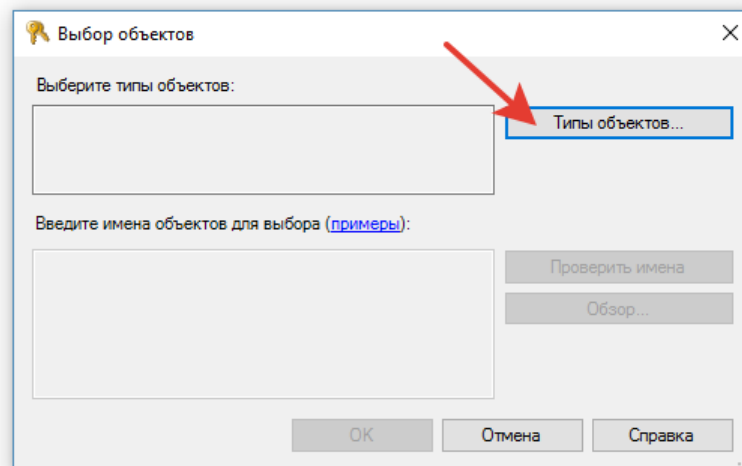
Перейдите на вкладку «**Защищаемые объекты**» и нажмите кнопку «**Найти**»:



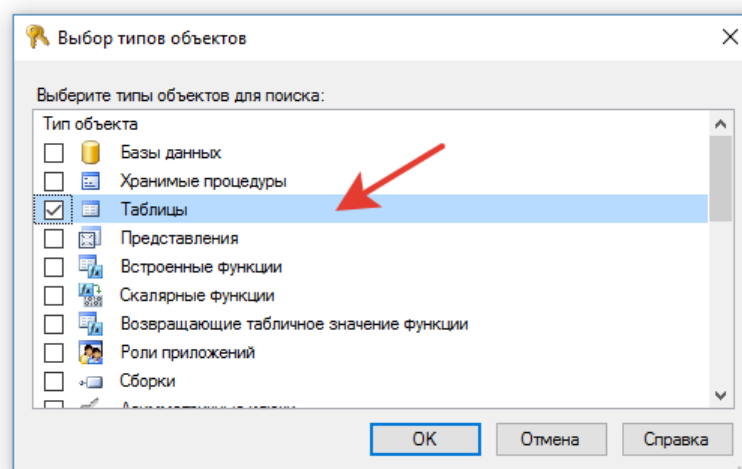
Выберите пункт «**Определенные объекты**» и нажмите кнопку «**OK**»:



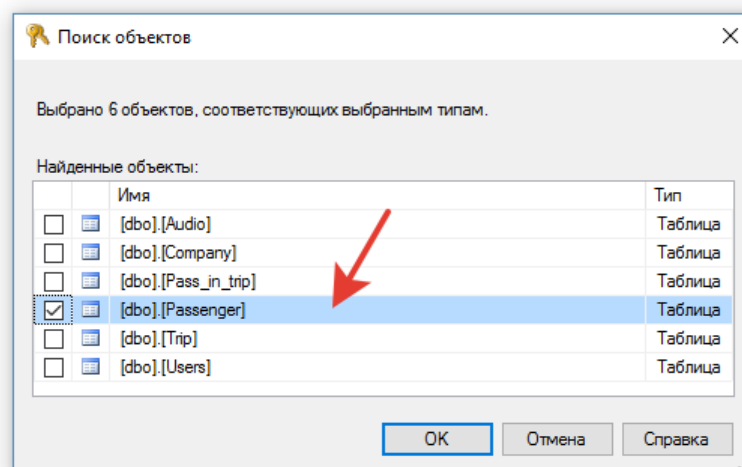
В открывшемся окне нажмите кнопку «**Типы объектов**»:



Поставьте галочку напротив пункта «**Таблицы**» и закройте окно, нажав кнопку «**ОК**»:

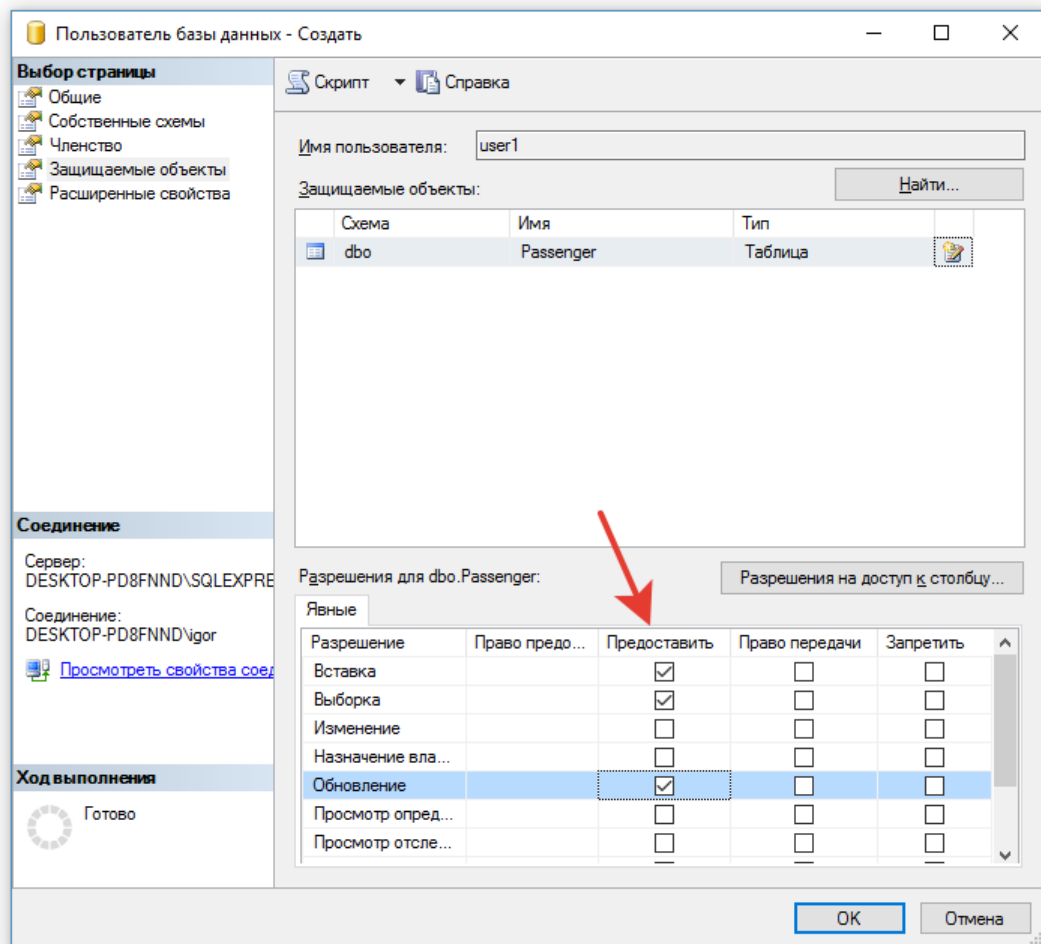


Нажмите кнопку «**Обзор**». В открывшемся окне поставьте галочку напротив таблицы «**[dbo].[Passenger]**» и нажмите кнопку «**ОК**»:



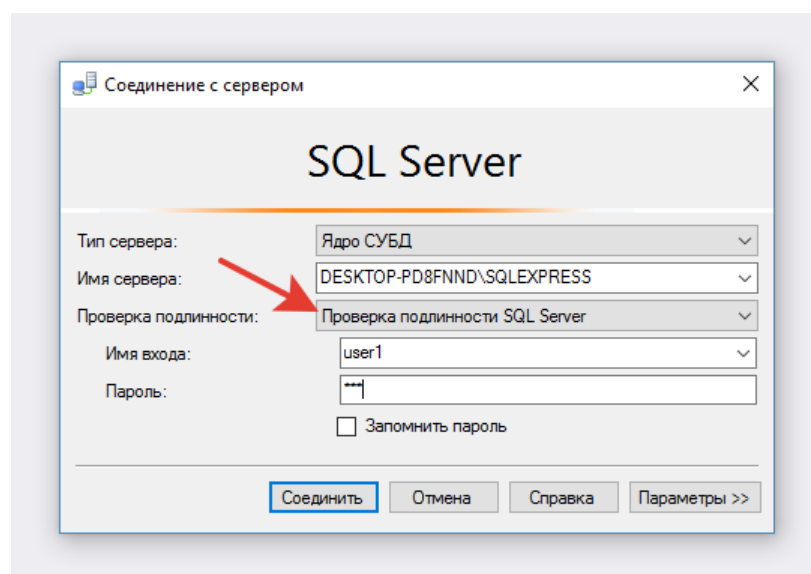
Еще раз нажмите кнопку «**ОК**». Мы добавили новое правило доступа.

Выберите наше единственное правило, нажав на нем левой кнопкой мыши. В панели снизу отобразится список доступных ограничений:

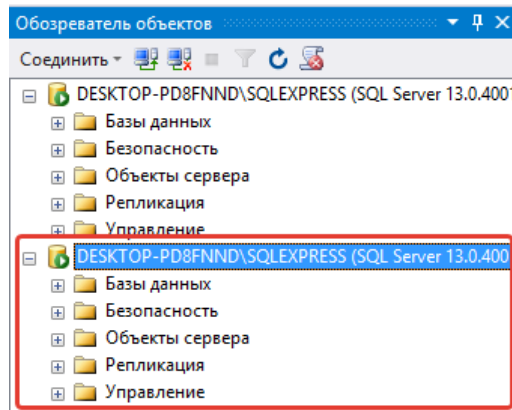


В столбце «Предоставить» установите галочки у строк «Вставка», «Выборка» и «Обновление». Теперь пользователь **user1** сможет выполнять с таблицей **Passenger** операции **SELECT**, **INSERT** и **UPDATE**. Другие операции, например добавление столбца в таблицу, ему будут недоступны.

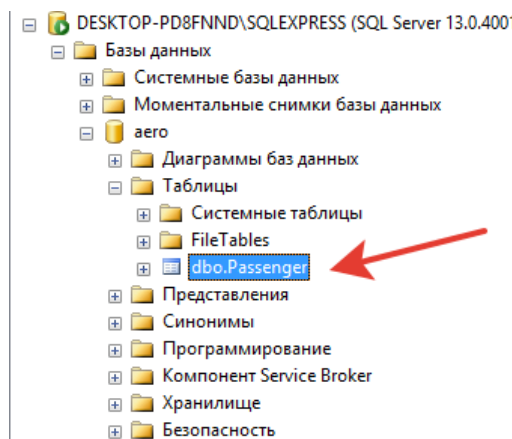
Можно проверить возможности нового пользователя из среды **SSMS**. В меню «Файл» выберите пункт «Подключить к обозревателю объектов». В окне подключения выберите новый способ проверки подлинности – «Проверка подлинности SQL Server». Введите имя пользователя «**user1**» и соответствующий ему пароль.



В окне обозревателя объектов появится новый узел сервера:

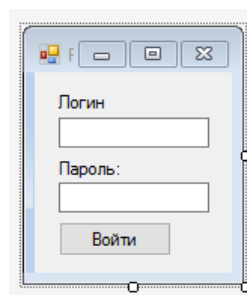


Если мы развернем узлы «Базы данных» - «aero» - «Таблицы», то увидим, что в данном подключении нам доступна всего одна таблица «Passenger»:



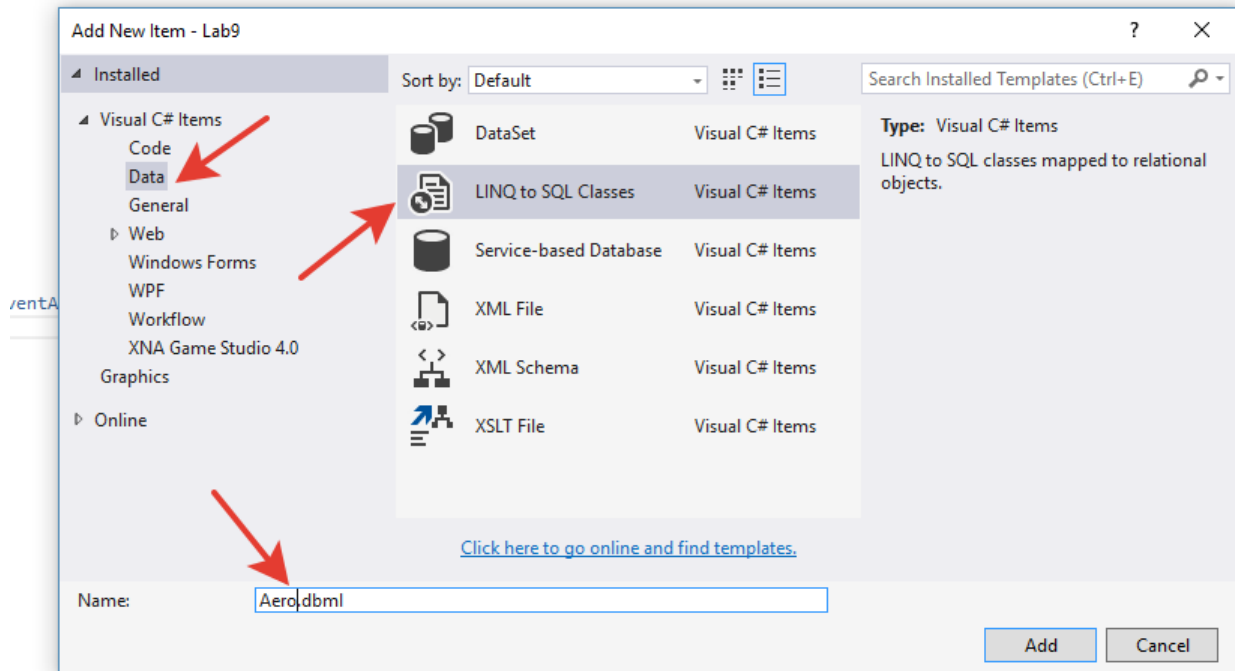
ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ ИЗ ПРИЛОЖЕНИЯ C# С ИСПОЛЬЗОВАНИЕМ ИМЕНИ ПОЛЬЗОВАТЕЛЯ И ПАРОЛЯ

Откройте **Visual Studio** и создайте новый проект **Windows Forms**. По аналогии с лабораторной работой №8 создайте форму следующего вида:



В свойствах кнопки «**Войти**» установите имя ее переменной равным «**connect**». Компонентам **TextBox** для ввода **логина** и **пароля** назначьте соответственно имена переменных «**login**» и «**password**». Дважды щелкните на кнопке «**Войти**» для создания обработчика события.

Добавьте в проект новый компонент **LINQ to SQL**, назовите его «**Aero**»:



Из панели **обозревателя серверов (Server Explorer)** перенесите все имеющиеся таблицы в окно конструктора.

Вернемся к коду формы.

Нам необходимо создать новое подключение к базе данных, используя в качестве параметров имя пользователя и пароль. Для это нам нужно указать эти параметры при создании нового объекта **DataContext** при помощи строки текста (по аналогии с лабораторной работой №2). Для облегчения процесса воспользуемся классом **SqlConnectionStringBuilder**.

```
var builder = new System.Data.SqlClient.SqlConnectionStringBuilder();
```

После создания объекта нам необходимо настроить параметры подключения. Указываем имя сервера:

```
builder["Server"] = "DESKTOP-PD8FNND\\SQLEXPRESS";
```

Устанавливаем начальную базу данных:

```
builder["Initial Catalog"] = "aero";
```

Задаем имя пользователя и его пароль:

```
builder.UserID = login.Text;  
builder.Password = password.Text;
```

Теперь создадим новое подключение к базе данных. Мне **не** будем использовать подключение **AeroDataContext** т.к. оно использует **пользователя Windows** для входа в базу данных, и поэтому не имеет ограничений в доступе к таблицам.

Нам необходимо создать подключение на основе базового класса **DataContext**:

```
var db = new System.Data.Linq.DataContext(builder.ConnectionString);
```

Для проверки корректности введенных данных вызовем функцию подключения к БД:

```
db.Connection.Open();
```

Метод Open выбросит исключение типа **SqlException** если при подключении возникают ошибки. Мы перехватим его и выведем сообщение о неудачном подключении. Полный код выглядит следующим образом:

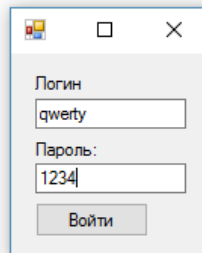
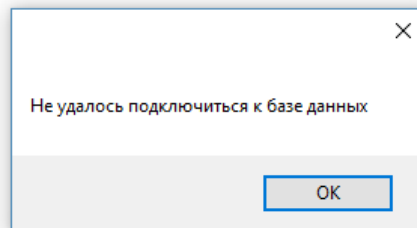
```
try {  
    db.Connection.Open();  
}
```

```

} catch (System.Data.SqlClient.SqlException) {
    MessageBox.Show("Не удалось подключиться к базе данных");
    return;
}

```

При вводе неверных логина или пароля мы получим сообщение об ошибке:

После успешного подключения можно начать работать с таблицами. К сожалению, базовый класс не содержит в себе ссылок на таблицы, поэтому нам придется получать их вручную:

```

var passengers = db.GetTable<Passenger>();

```

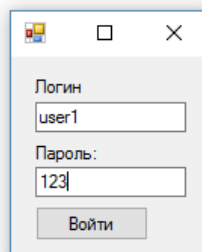
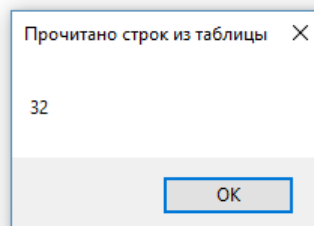
После получения ссылки на таблицу мы можем выполнять к ней запросы:

```

var query = from c in passengers
            select c;
MessageBox.Show(query.Count().ToString(), "Прочитано строк из таблицы");

```

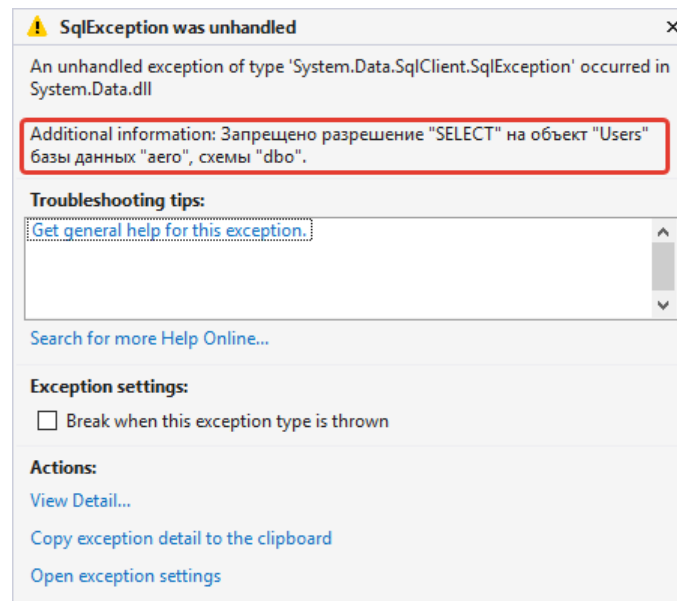
При успешном подключении к БД мы увидим сообщение с количеством успешно выбранных строк:

Если мы попробуем прочитать таблицу, отличную от **Passengers**, то вместо результата будет выброшено исключение с ошибкой доступа:


```
var users = db.GetTable<User>();  
var query = from c in users  
            select c;  
MessageBox.Show(query.Count().ToString(), "Прочитано строк из таблицы");
```

Результат:



ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

При попытке чтения данных из таблицы **Users** выбрасывается исключение и программа вылетает. Добавьте код обработки ошибки, чтобы вместо аварийного завершения работы программа выводила сообщение «Ошибка базы данных».