

Лабораторная работа №17-18. Работа с базами данных в WPF

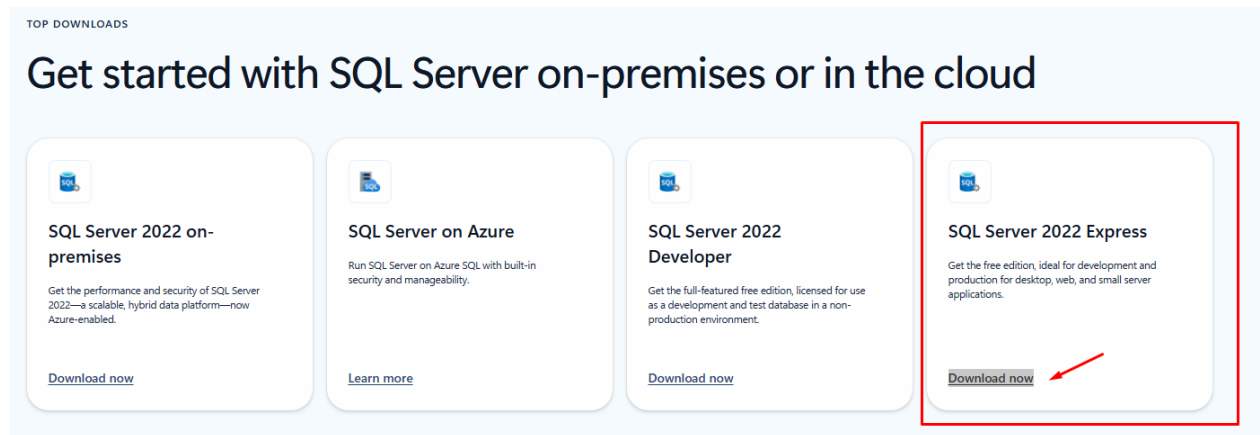
Цель: Научиться подключать и работать с MS SQL Server, а также создать рабочее приложение в WPF.

Шаг 1. Скачивание SQL Server

1. Перейдите на сайт:

<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

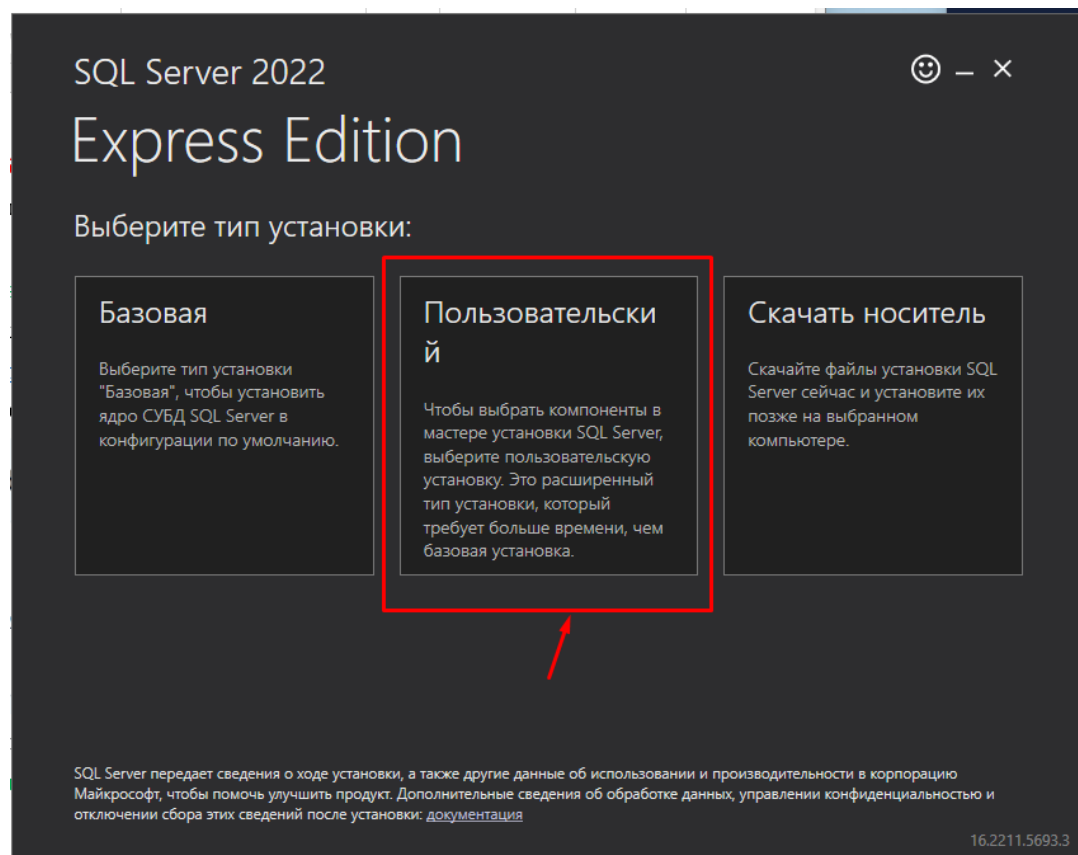
3. Найдите версию **SQL Server 2022 Express** и нажмите **Download now**.



Шаг 2. Запуск установщика

3. После скачивания запустите установщик.

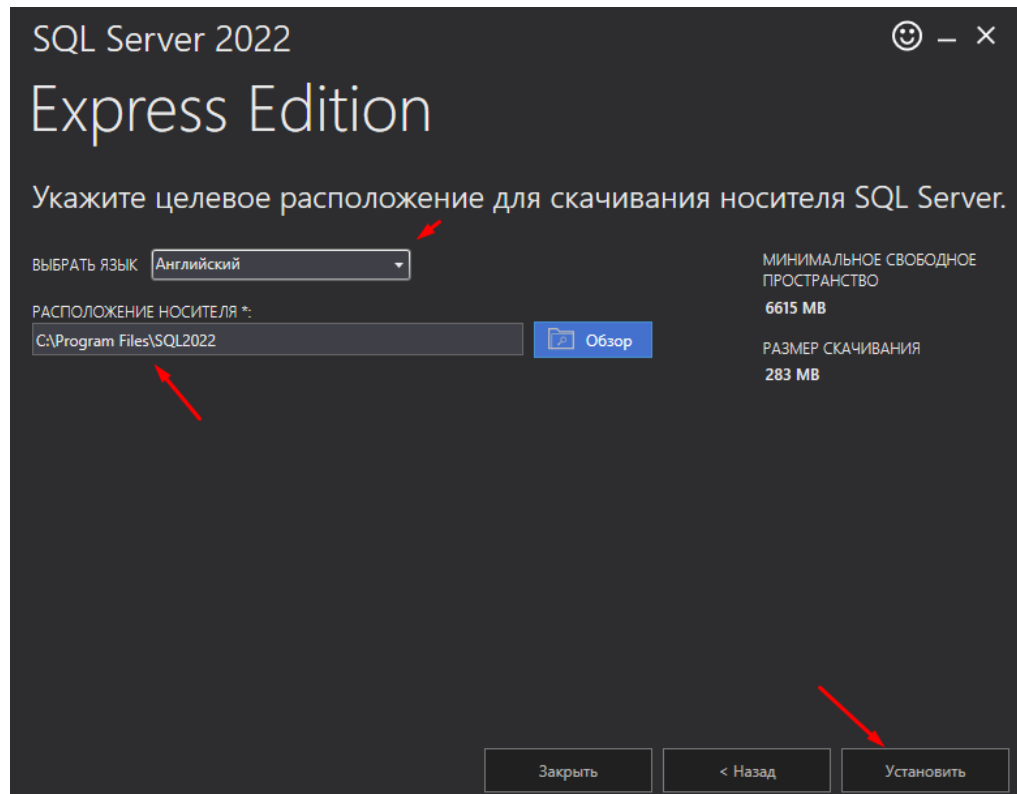
4. Выберите **пользовательский тип установки (Custom)**.



5. Установите язык на **English**, папку установки задайте:

C:\Program Files\SQL2022

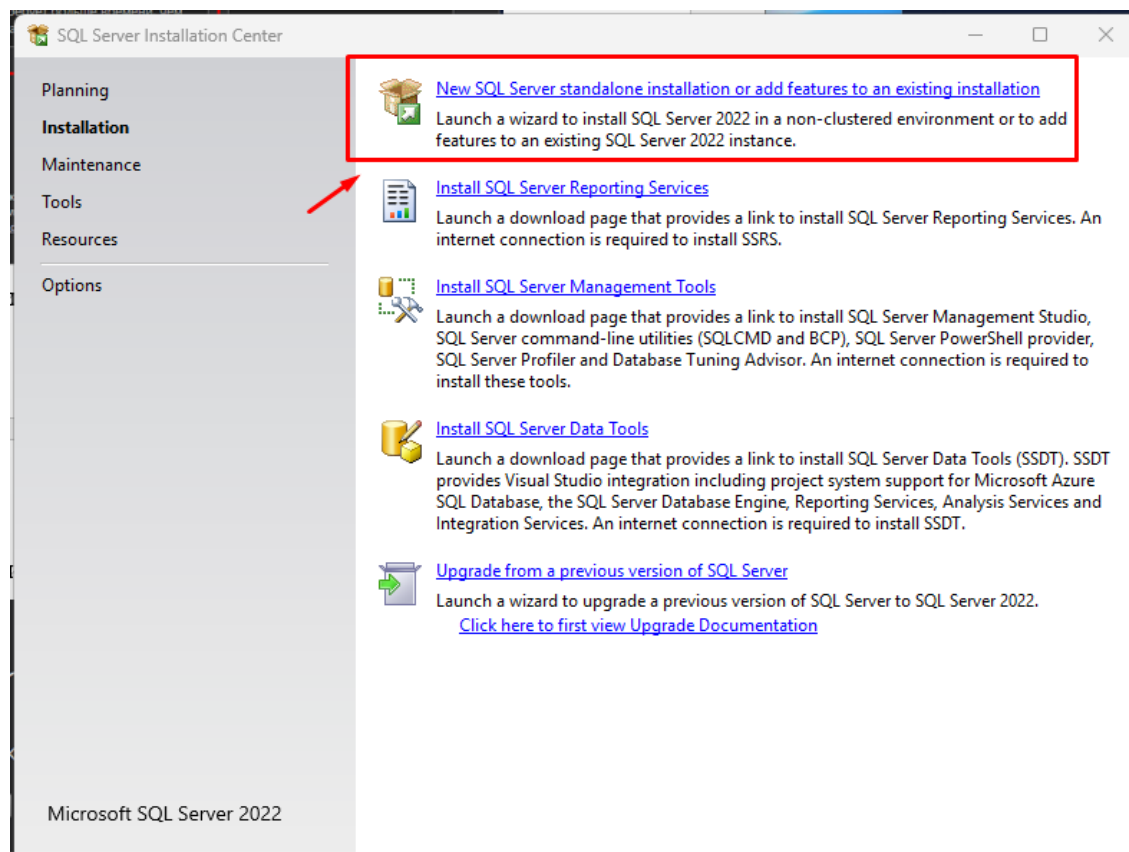
Нажмите **Установить** и дождитесь завершения установки.



Дождитесь окончания установки.

Шаг 3. Установка SQL Server

6. При запуске лаунчера выберите **New SQL Server standalone installation**.



7. Нажмите Next.

Microsoft Update
Use Microsoft Update to check for important updates

Global Rules

Microsoft Update

Product Updates

Install Setup Files

Install Rules

Installation Type

License Terms

Azure Extension for SQL Server

Feature Selection

Feature Rules

Feature Configuration Rules

Installation Progress

Complete

Microsoft Update offers security and other important updates for Windows and other Microsoft software, including SQL Server 2022. Updates are delivered using Automatic Updates, or you can visit the Microsoft Update website.

☐ Use Microsoft Update to check for updates (recommended)

[Microsoft Update FAQ](#)

[Microsoft Privacy Statement](#)

< Back

Next >

Cancel

8. Дождитесь загрузки обновлений (если обновления Windows отключены — может появиться ошибка). Нажмите Next.

Product Updates
Always install the latest updates to enhance your SQL Server security and performance.

Global Rules

Microsoft Update

Product Updates

Install Setup Files

Install Rules

Installation Type

License Terms

Azure Extension for SQL Server

Feature Selection

Feature Rules

Feature Configuration Rules

Installation Progress

Complete

✔ There are no updates for SQL Server found online.

[Read our privacy statement online](#)

[Learn more about SQL Server product updates](#)

< Back

Next >

Cancel

9. Нажмите Next.

Installation Type

Perform a new installation or add features to an existing instance of SQL Server 2022.

- Global Rules
- Microsoft Update
- Product Updates
- Install Setup Files
- Install Rules
- Installation Type**
- License Terms
- Azure Extension for SQL Server
- Feature Selection
- Feature Rules
- Instance Configuration
- Server Configuration
- Database Engine Configuration
- Feature Configuration Rules
- Installation Progress
- Complete

☒ **Perform a new installation of SQL Server 2022**

Select this option if you want to install a new instance of SQL Server or want to install shared components.

☐ **Add features to an existing instance of SQL Server 2022**

SQLEXPRESS

Select this option if you want to add features to an existing instance of SQL Server. For example, you want to add the Analysis Services features to the instance that contains the Database Engine. Features within an instance must be the same edition.

Installed instances:

Instance Name	Instance ID	Features	Edition	Version
SQLEXPRESS	MSSQL16.SQLEXPRESS	SQLEngine	Express	16.0.1000.6

< Back

Next >

Cancel

10. Согласитесь с лицензией (I accept the license terms) → Next.

License Terms

To install SQL Server 2022, you must accept the Microsoft Software License Terms.

- Global Rules
- Microsoft Update
- Product Updates
- Install Setup Files
- Install Rules
- Installation Type
- License Terms**
- Azure Extension for SQL Server
- Feature Selection
- Feature Rules
- Instance Configuration
- Server Configuration
- Database Engine Configuration
- Feature Configuration Rules
- Installation Progress
- Complete

SQL Server 2022 Express Edition

YOU MUST ACCEPT THE SOFTWARE LICENSE TERMS. SEE BELOW. Please read the full license terms provided at (aka.ms/useterms).

DATA COLLECTION. The software may collect information about you and your use of the software and send that to Microsoft. Microsoft may use this information to provide services and improve Microsoft's products and services. Your opt-out rights, if any, are described in the product documentation. Some features in the software may enable collection of data from users of your applications that access or use the software. If you use these features to enable data collection in your applications, you must comply with applicable law, including getting any required user consent, and maintain a prominent privacy policy that accurately informs users about how you use, collect, and share their data. You can learn more about Microsoft's data collection and use in the product documentation and the Microsoft Privacy Statement at <https://go.microsoft.com/fwlink/?LinkId=521829>. You agree to comply with all

☒ I accept the license terms and [Privacy Statement](#)

SQL Server transmits information about your installation experience as well as other usage and performance data. Azure Arc connection also transmits the configuration data to allow you to manage and protect your SQL Server instance using Azure Portal and services. To learn more about data processing and privacy controls, and to turn off the collection of certain information, see the [documentation](#).

Copy

Print

< Back

Next >

Cancel

11. Снимите галочку с **Azure Extension** → **Next**.

Azure Extension for SQL Server

Azure Extension for SQL Server is required to enable Microsoft Defender for Cloud, Purview, and Azure Active Directory.

Global Rules

Microsoft Update

Product Updates

Install Setup Files

Install Rules

Installation Type

License Terms

Azure Extension for SQL Serv...

Feature Selection

Feature Rules

Instance Configuration

Server Configuration

Database Engine Configuration

Feature Configuration Rules

Installation Progress

Complete

☐ Azure Extension for SQL Server

To install Azure extension for SQL Server, provide your Azure account or a service principal to authenticate the SQL Server instance to Azure. You also need to provide the Subscription ID, Resource Group, Region, and Tenant ID where this instance will be registered. For more information for each parameter, visit <https://aka.ms/arc-sql-server>.

☐ Use Azure Login

☒ Use Service Principal

Azure Service Principal ID*

Azure Service Principal Secret*

Azure Subscription ID*

Azure Resource Group*

Azure Region*

Azure Tenant ID*

Proxy Server URL (optional)

< Back

Next >

Cancel

12. Снимите галочку с **Machine Learning Services and Language Extensions** (не требуется) → **Next**.

Feature Selection

Select the Express features to install.

Global Rules

Microsoft Update

Product Updates

Install Setup Files

Install Rules

Installation Type

License Terms

Azure Extension for SQL Server

Feature Selection

Feature Rules

Instance Configuration

Server Configuration

Database Engine Configuration

Feature Configuration Rules

Installation Progress

Complete

Looking for Reporting Services? [Download it from the web](#)

Features:

Instance Features

☒ Database Engine Services

☒ SQL Server Replication

☒ Machine Learning Services and Language Extensions

☒ Full-Text and Semantic Extractions for Search

☐ PolyBase Query Service for External Data

Shared Features

☐ LocalDB

Redistributable Features

Select All

Unselect All

Instance root directory: C:\Program Files\Microsoft SQL Server\

Shared feature directory: C:\Program Files\Microsoft SQL Server\

Shared feature directory (x86): C:\Program Files (x86)\Microsoft SQL Server\

Feature description:

Includes extensions that enable integration with R, Python, Java and other programming languages using standard T-SQL statements.

Prerequisites for selected features:

Already installed:

Windows PowerShell 3.0 or higher

Microsoft Visual C++ 2017 Redistributable

Disk Space Requirements

Drive C: 1415 MB required, 209972 MB available

< Back

Next >

Cancel

13. Оставьте имя экземпляра по умолчанию **MSSQLSERVER** → Next.

Instance Configuration

Specify the name and instance ID for the instance of SQL Server. Instance ID becomes part of the installation path.

Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Installation Type
License Terms
Azure Extension for SQL Server
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Installation Progress
Complete

☒ Default instance
☐ Named instance: * MSSQLSERVER

Instance ID: MSSQLSERVER

SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER

Installed instances:

Instance Name	Instance ID	Features	Edition	Version
SQLEXPRESS	MSSQL16.SQLEXPRESS	SQLEngine	Express	16.0.1000.6

< Back Next > Cancel

14. Выберите **SQL Server Database Engine** → Next.

Server Configuration

Specify the service accounts and collation configuration.

Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Installation Type
License Terms
Azure Extension for SQL Server
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Installation Progress
Complete

Service Accounts Collation

Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Database Engine	NT Service\MSSQLSERVER		Automatic
SQL Full-text Filter Daemon Launch...	NT Service\MSSQLFDLa...		Manual
SQL Server Browser	NT AUTHORITY\LOCAL...		Disabled

☐ Grant Perform Volume Maintenance Tasks privilege to SQL Server Database Engine Service

This privilege enables instant file initialization by avoiding zeroing of data pages. This may lead to information disclosure by allowing deleted content to be accessed.

[Click here for details](#)

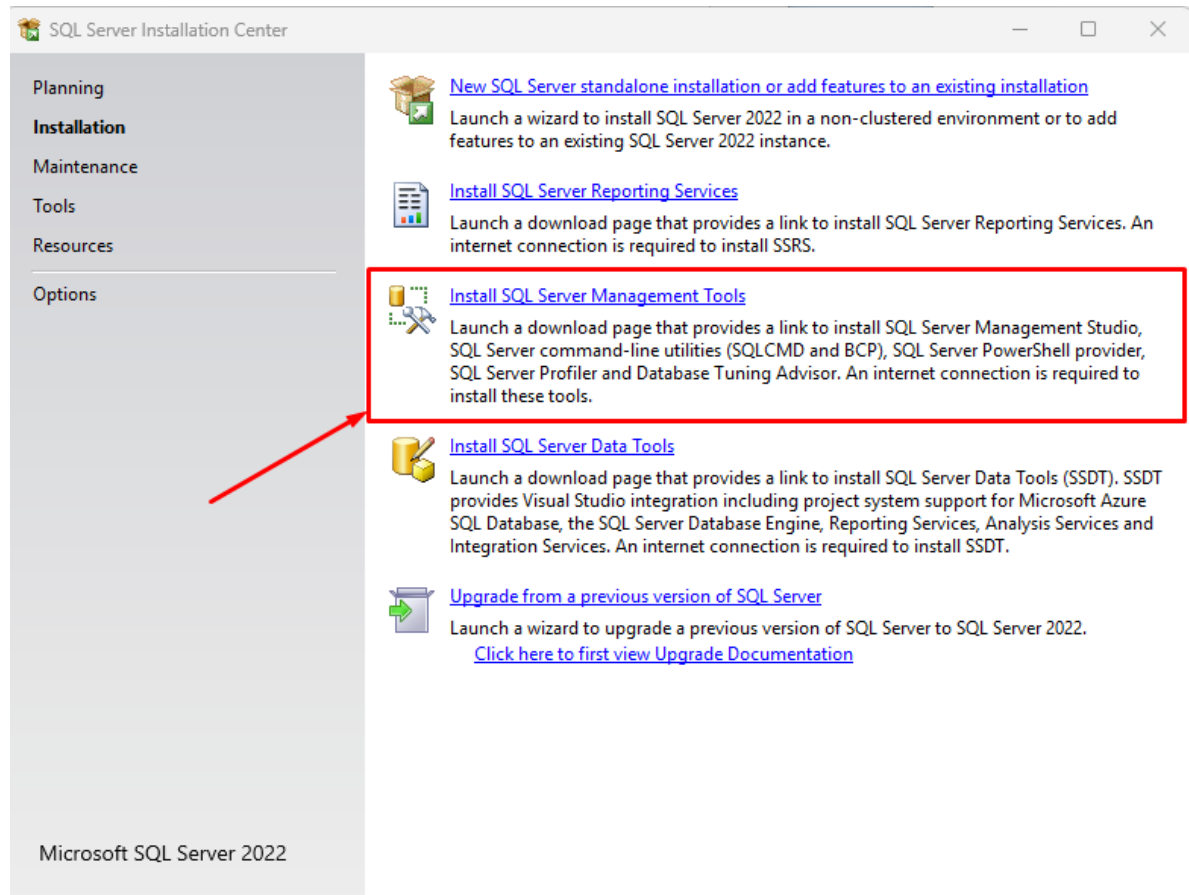
< Back Next > Cancel

15. В режиме аутентификации выберите **Mixed Mode**:

Шаг 4. Установка SQL Server Management Studio (SSMS)

Мы с вами установили SQL Server, но он не содержит графический интерфейс.

1. В лаунчере выберите пункт **SQL Server Management Tools** — откроется сайт Microsoft.



2. У вас перейдёт на сайт Microsoft. Найдите и скачайте **SSMS 21**.

Шаг 2. Определение версии SQL Server Management Studio для установки

Выберите версию SSMS для установки. Наиболее распространенными вариантами являются:

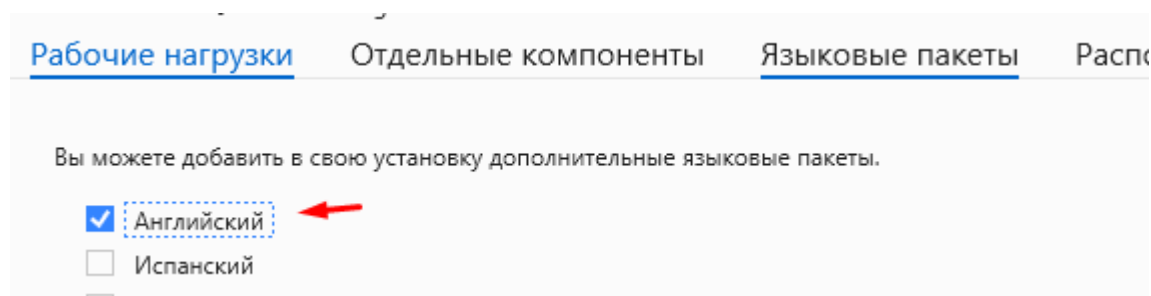
- Последний выпуск SQL Server Management Studio 21, размещенный на серверах Майкрософт. Чтобы установить эту версию, выберите следующую ссылку. Установщик загружает небольшой *загрузчик* в папку *Загрузки*.

[Скачивание SSMS 21 ↗](#)

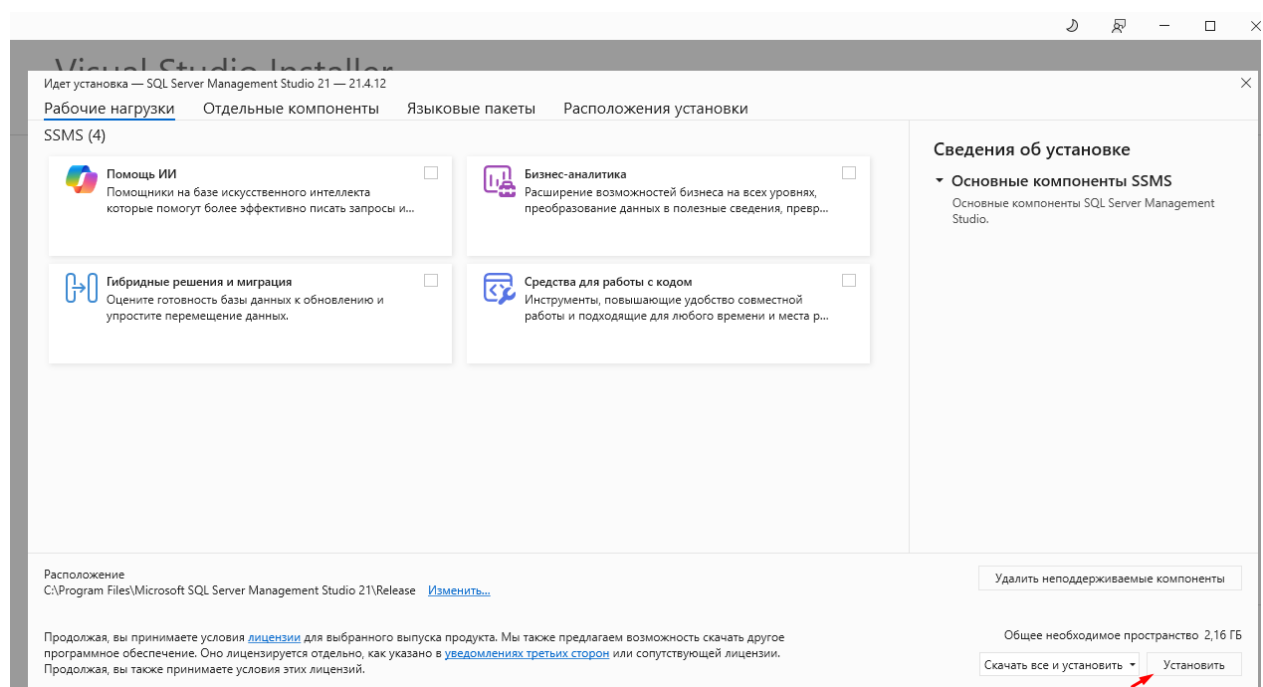
- Если у вас уже установлен SQL Server Management Studio 21, вы можете [установить другую версию вместе с ней](#).
- Вы можете скачать загрузчик или установщик для определенной версии на [странице истории выпусков](#) и использовать его для установки иной версии SSMS.

3. Запустите установщик SSMS.

4. В языковых компонентах поставьте галочку на **Английский**:



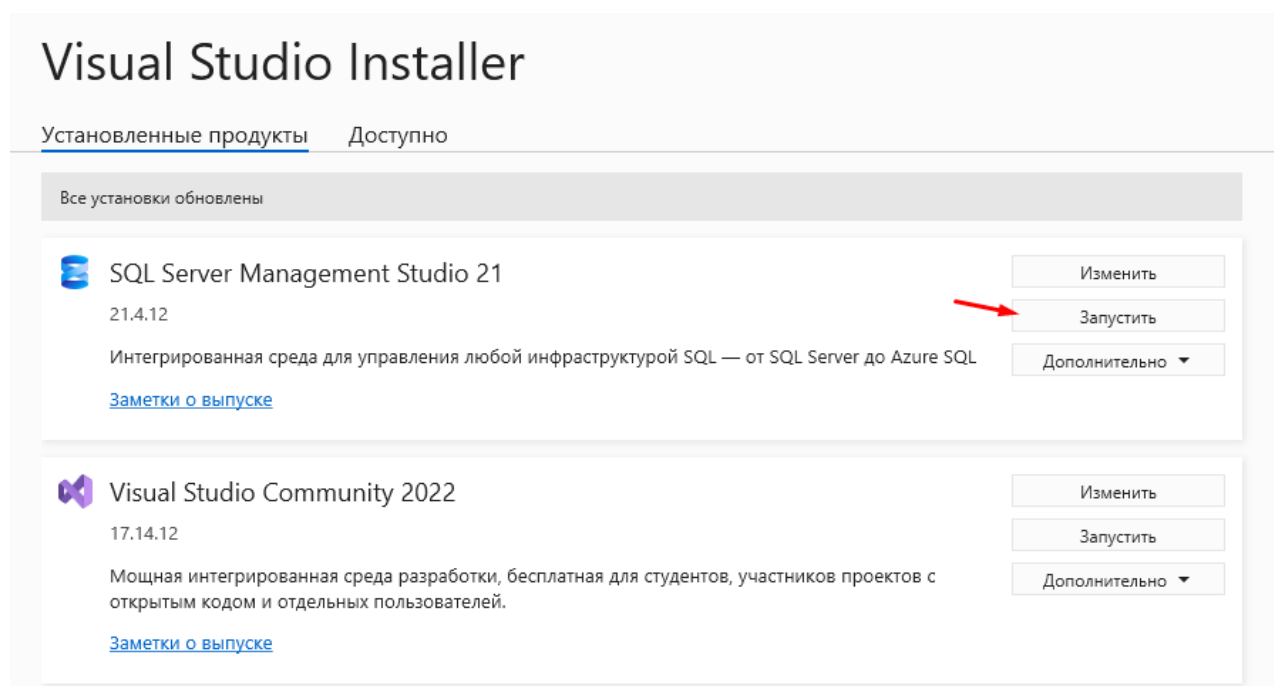
Основные настройки оставьте по умолчанию и нажмите **Установить**.



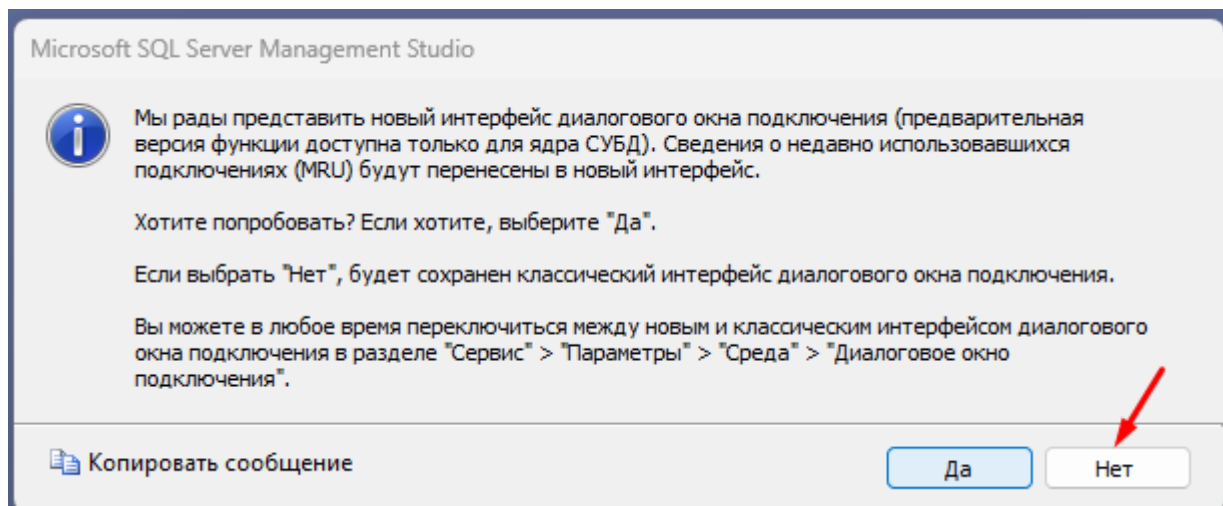
Шаг 5. Настройка SQL Server

1. Запуск SSMS

1. Запустите **SQL Server Management Studio 21**.

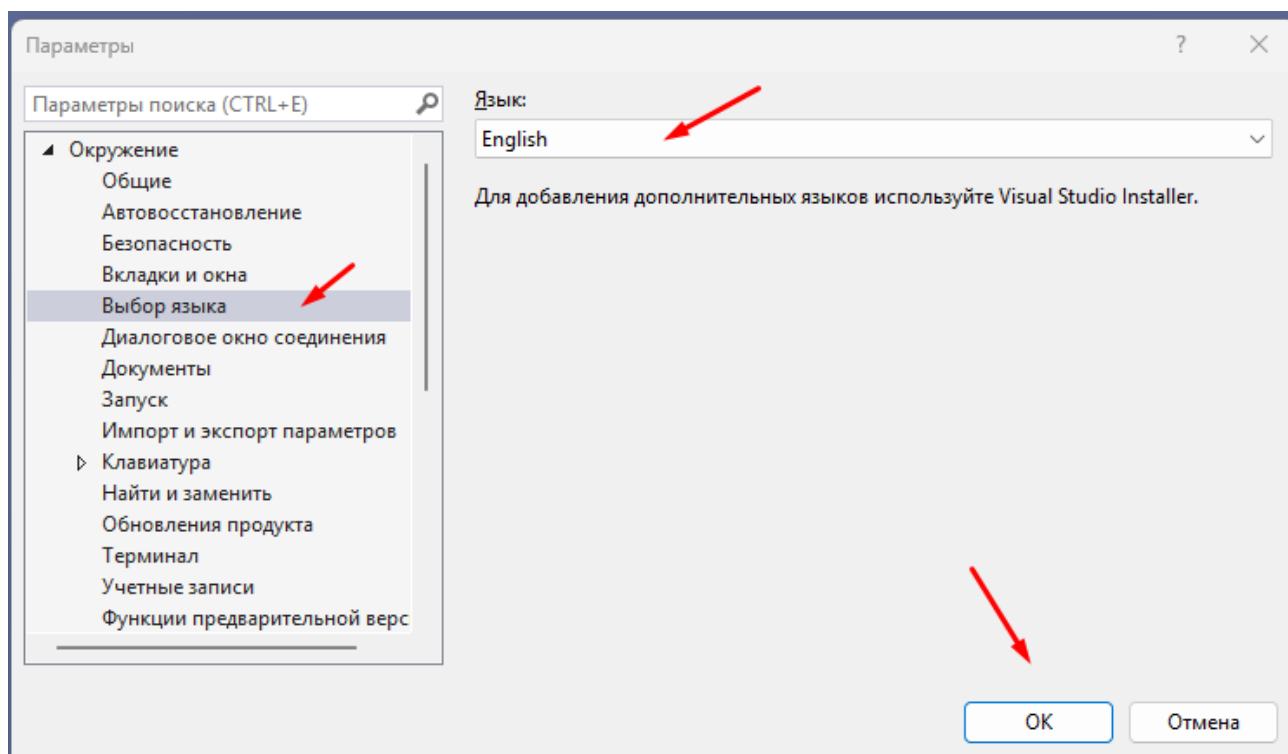


2. Если появится окно с предложением попробовать новый интерфейс — нажмите **Нет**.



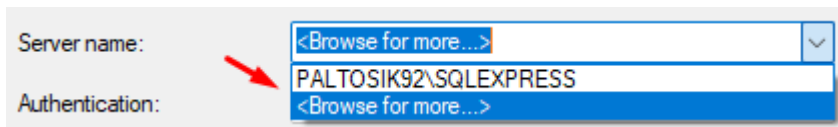
3. Изменение языка

- Закройте окно подключения к серверу.
- Перейдите в меню **Tools** → **Options**.
- В разделе настроек смените язык на **English**.
- Перезапустите SSMS, чтобы изменения вступили в силу.

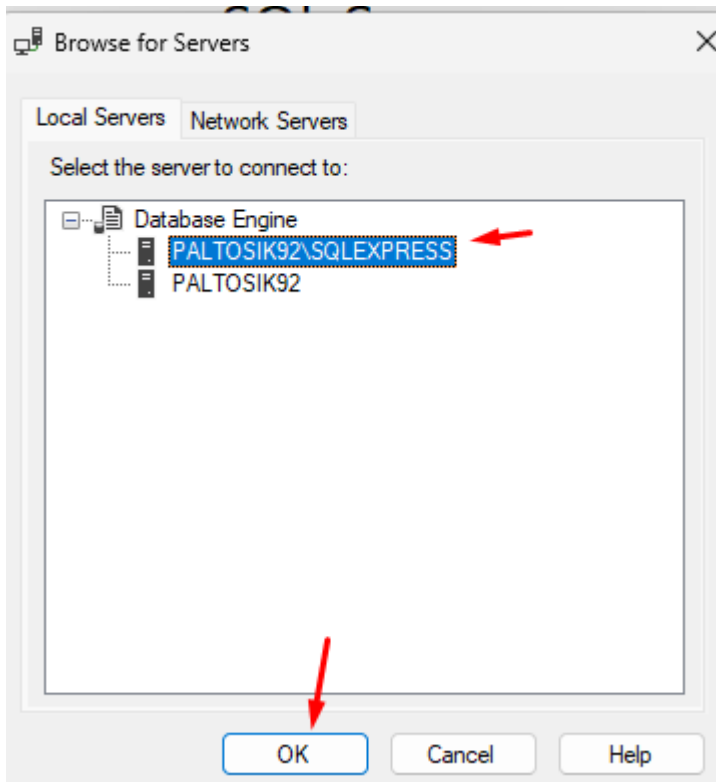


4. При повторном запуске, в окне подключения (**Connect to Server**) убедитесь, что в поле **Authentication** выбран правильный пользователь.

- Если пользователь не выбран, нажмите **Browse for more**.

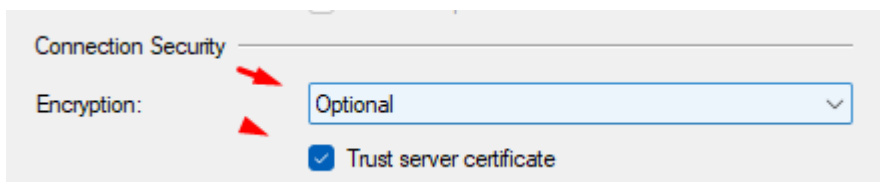


- Выберите своего пользователя с припиской /<имя_пользователя>/SQLEXPRESS.

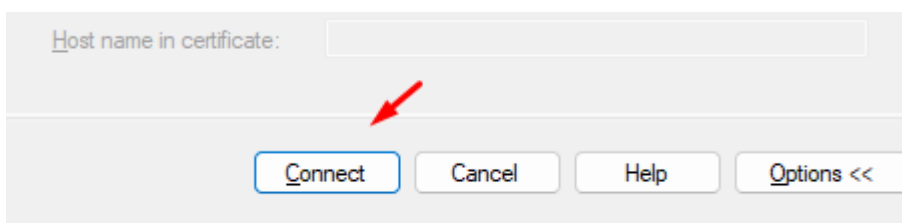


5. В блоке **Connection Security**:

- Установите **Encryption** в значение **Optional**.
- Поставьте галочку **Trust server certificate**.



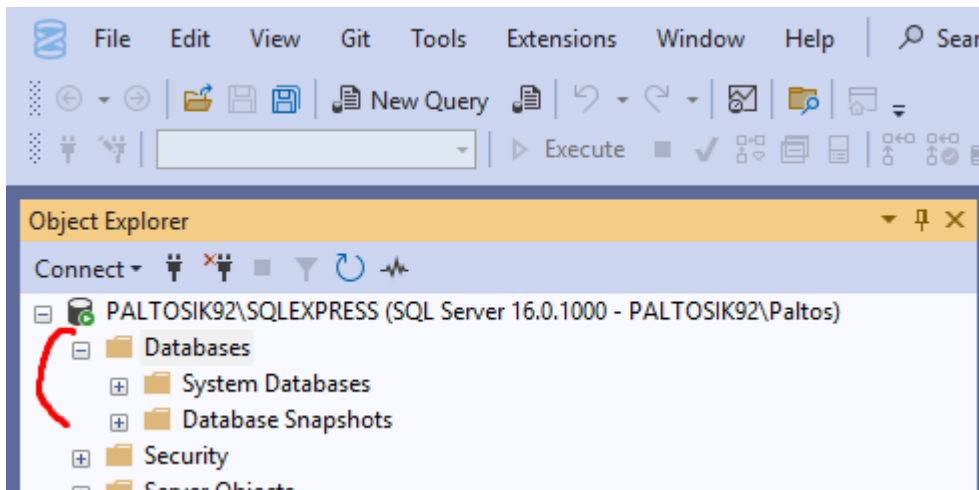
6. Нажмите **Connect**, чтобы подключиться к серверу.



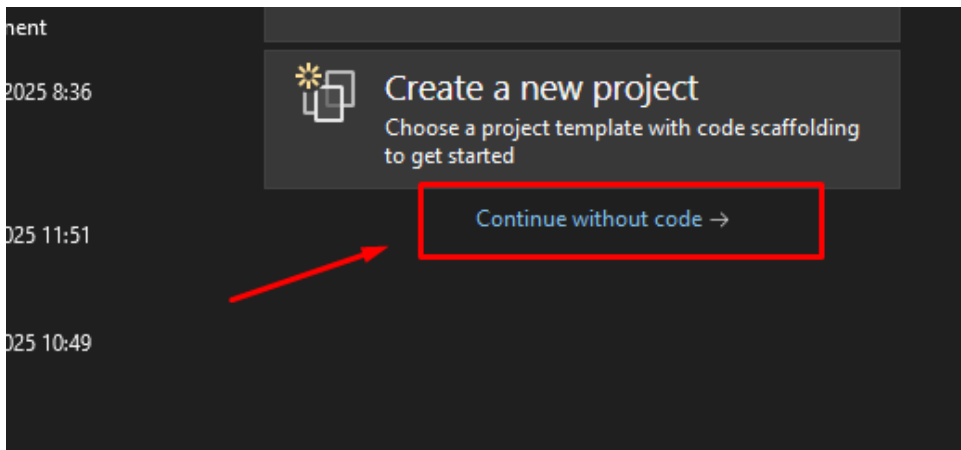
Шаг 6. Работа с SQL Server через Visual Studio

1. В SQL Server Management Studio (SSMS) откройте папку **Databases**.

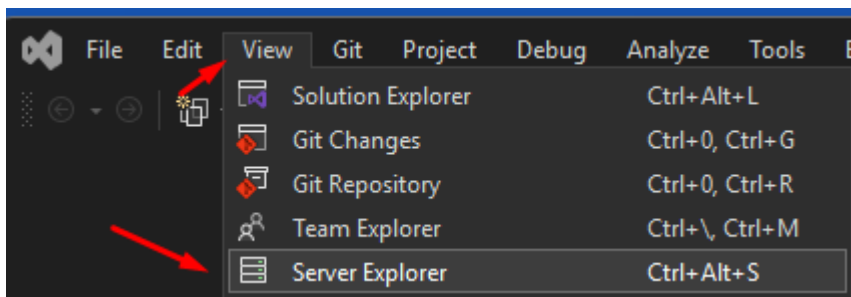
- Обратите внимание: пользовательские базы данных сейчас отсутствуют.



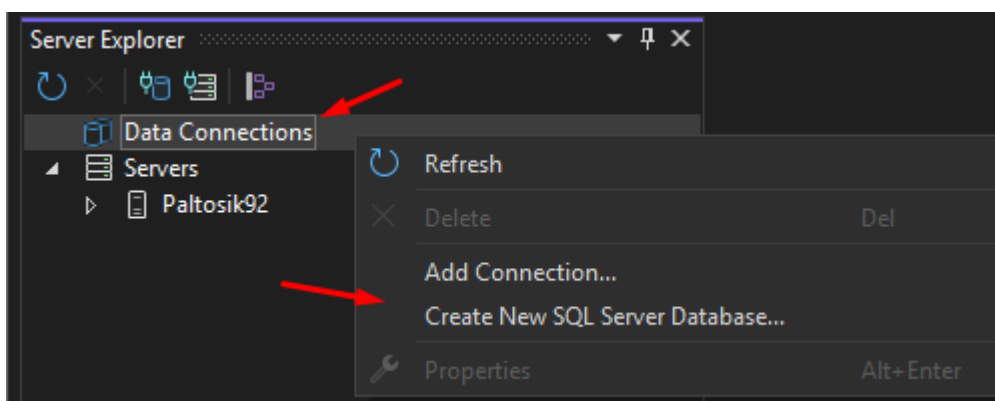
2. Запустите **Visual Studio**. В окне приветствия выберите **Continue without code**.



В меню Visual Studio выберите **View** → **Server Explorer**.



В панели **Server Explorer** щёлкните правой кнопкой мыши по **Data Connections** и выберите **Create New SQL Server Database**.



В первое поле нам нужно с вами добавить имя нашего сервера:

Enter information to connect to a SQL Server, then specify the name of a database to create.

Server name:

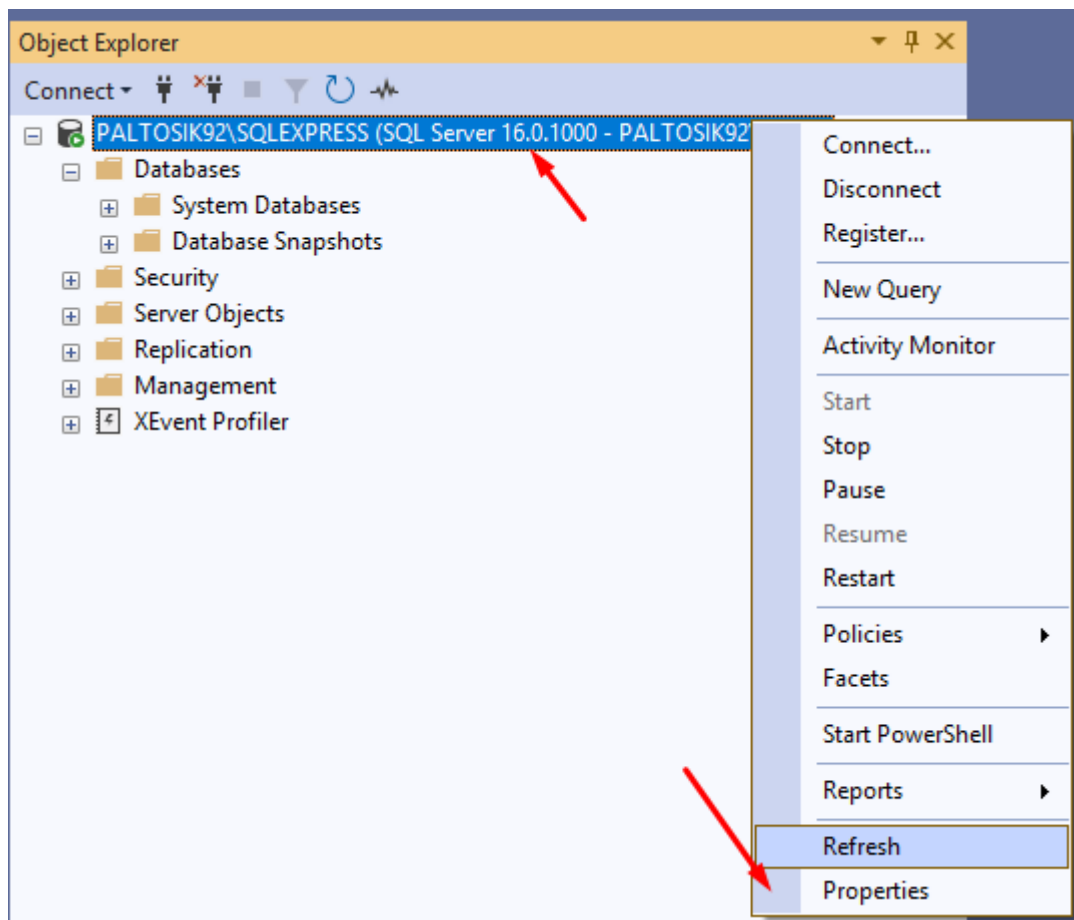
Refresh

Log on to the server

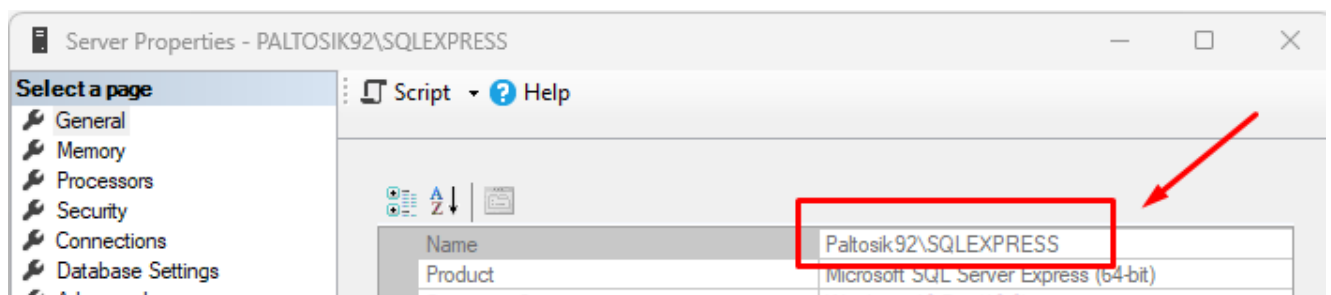
☒ Use Windows Authentication

3. Получение имени сервера

Перейдите в **SSMS**. Щёлкните правой кнопкой мыши по имени сервера → выберите **Properties**.



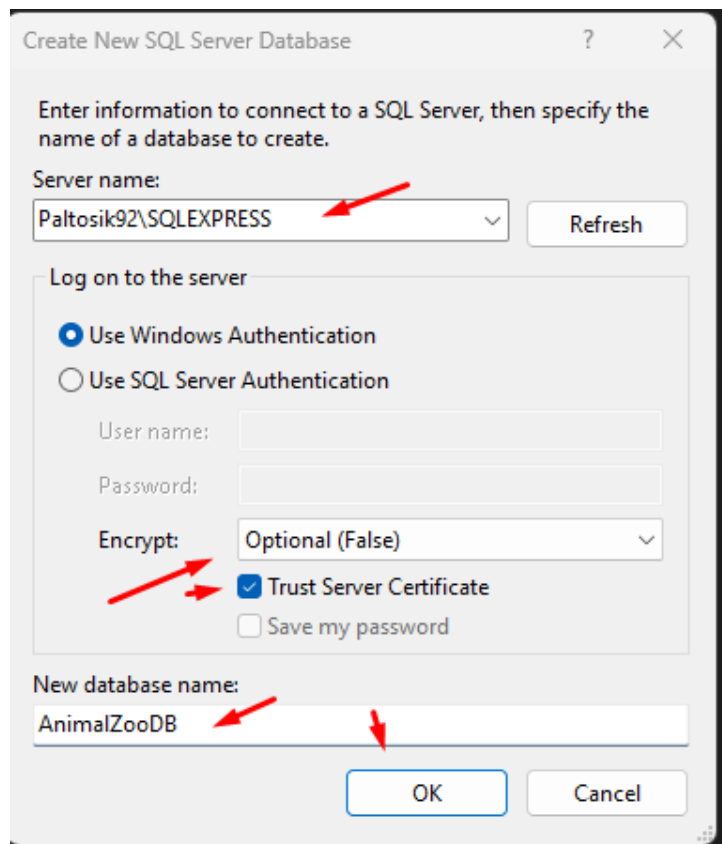
Из поля **Name** скопируйте имя сервера.



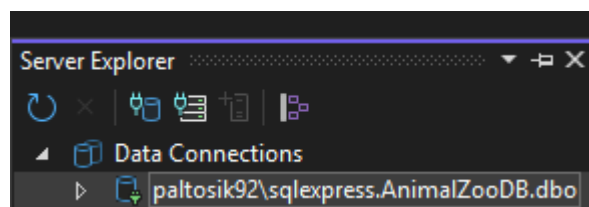
4. Вернитесь в окно **Create New SQL Server Database** в Visual Studio.

- Вставьте скопированное имя сервера в первое поле.
- Измените **Encrypt** на **Optional (False)**.

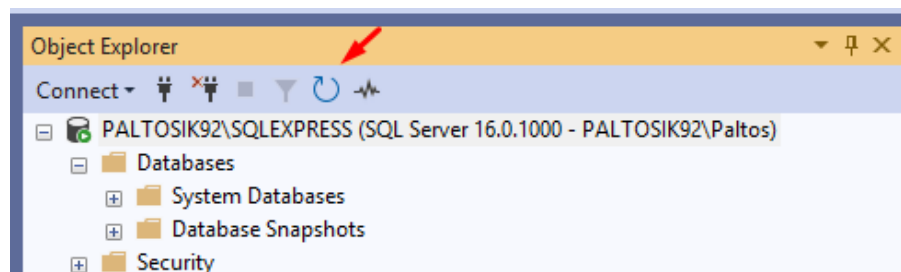
- Поставьте галочку **Trust Server Certificate**.
- Введите имя новой тестовой базы данных — **AnimalZooDB**.
- Нажмите **ОК**.



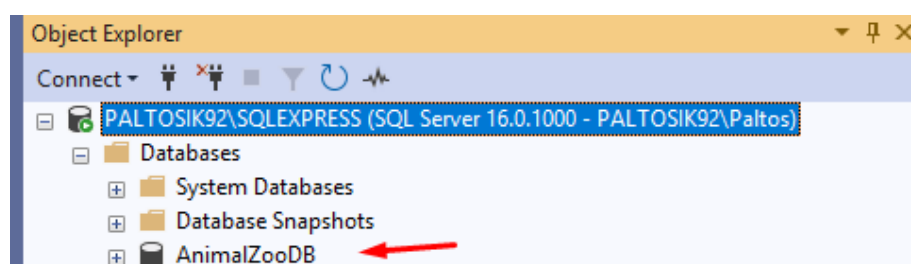
В **Visual Studio** в разделе **Data Connections** появится новая база данных.



Перейдите в **SSMS** и нажмите **Refresh** на списке баз данных.



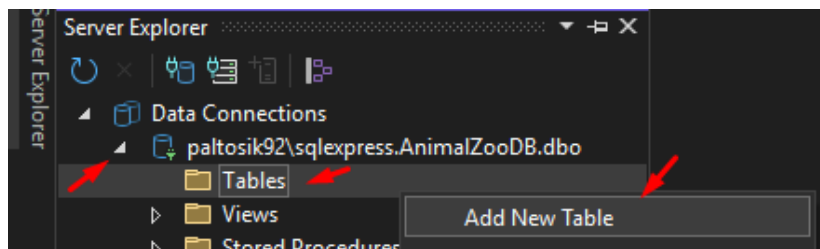
Убедитесь, что база **AnimalZooDB** появилась.



Шаг 7. Создание таблицы и подключение базы данных к WPF

1. Создание таблицы в базе данных

1. В **Visual Studio** раскройте свою базу данных.
2. Перейдите в папку **Tables**, щёлкните **ПКМ** → **Add New Table**.




2. Структура таблицы

В открывшемся окне вы увидите:


- **Имя столбца (Name)** — название поля таблицы.
- **Тип данных (Data Type)** — определяет формат данных.
- **Allow Nulls** — разрешает хранить NULL (пустое значение).
- **Default** — значение по умолчанию.

Слева от имени столбца **ключик** означает, что этот столбец является **ключевым** (Primary Key).


	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
			<input type="checkbox"/>	

3. Добавление нового столбца

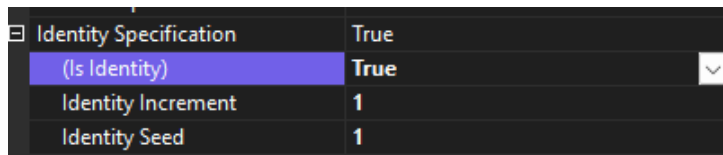
1. Создайте новый столбец:
 - **Name:** Location
 - **Data Type:** NVARCHAR(50) (строка до 50 символов, поддерживает Unicode)
 - Снимите галочку **Allow Nulls** (обязательное заполнение).

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	Location	nvarchar(50)	<input type="checkbox"/>	

2. Выберите столбец **Id** и нажмите **F4** для открытия окна **Properties**.

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	Location	nvarchar(50)	<input type="checkbox"/>	

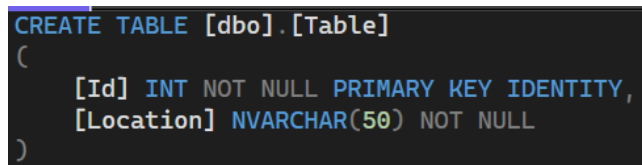
3. Найдите параметр **Identity Specification** → **Is Identity** и установите его в **True**.
- Это нужно, чтобы Id автоматически увеличивался при добавлении новых записей.



Identity Specification	True
(Is Identity)	True
Identity Increment	1
Identity Seed	1

4. Автоматически сгенерированный SQL

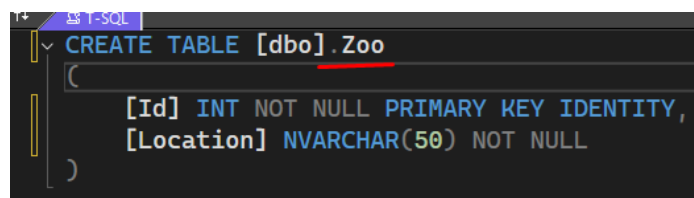
Внизу окна вы увидите SQL-запрос, соответствующий вашей таблице:



```
CREATE TABLE [dbo].[Table]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Location] NVARCHAR(50) NOT NULL
)
```

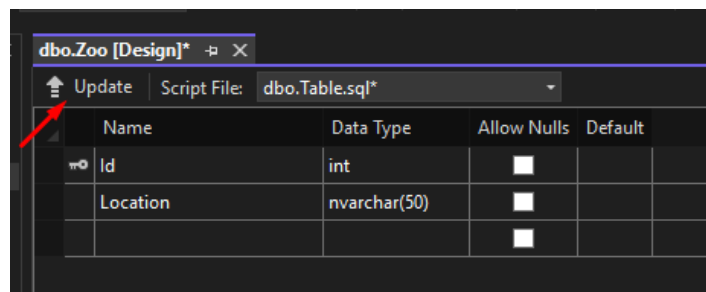
5. Сохранение и обновление таблицы

1. Переименуйте таблицу в **Zoo**.

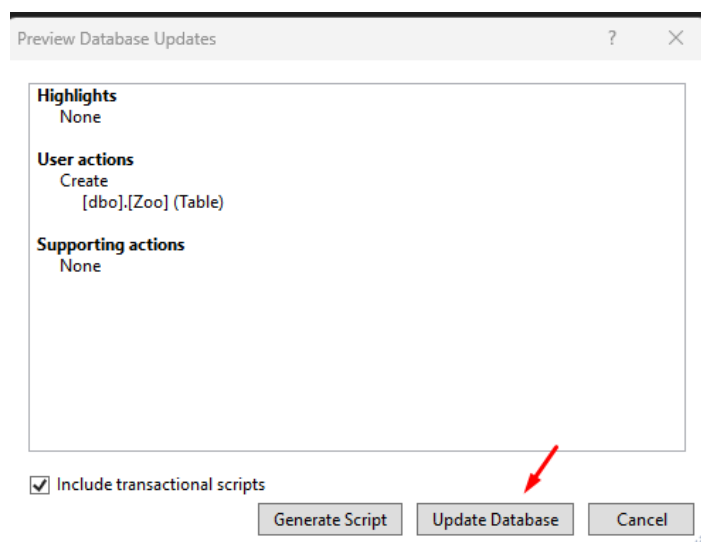


```
CREATE TABLE [dbo].[Zoo]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Location] NVARCHAR(50) NOT NULL
)
```

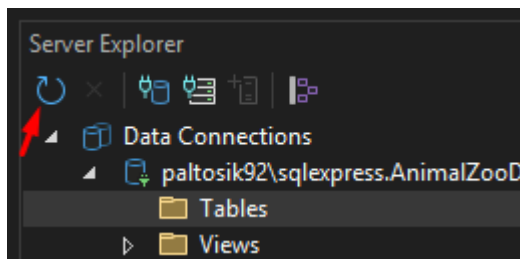
2. Нажмите кнопку **Update**.



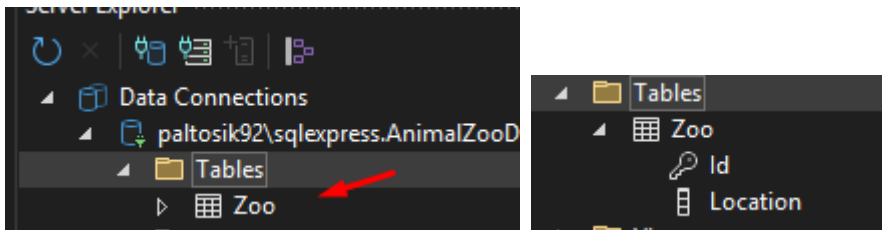
3. В появившемся окне выберите **Update Database**.



4. Нажмите **Refresh** в списке таблиц базы данных.

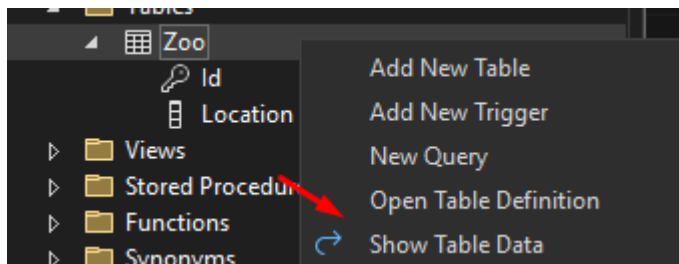


5. Убедитесь, что появилась таблица Zoo со столбцами Id и Location.

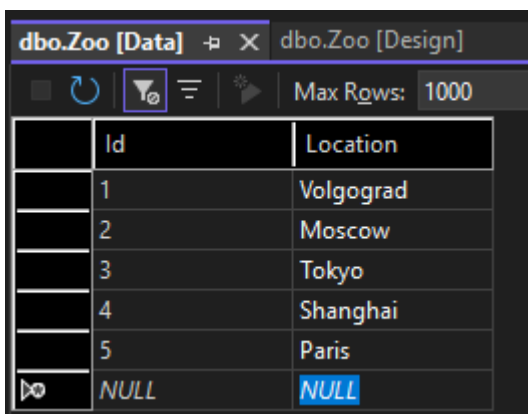


6. Заполнение таблицы

1. Щёлкните ПКМ по таблице **Zoo** и выберите **Show Table Data**.

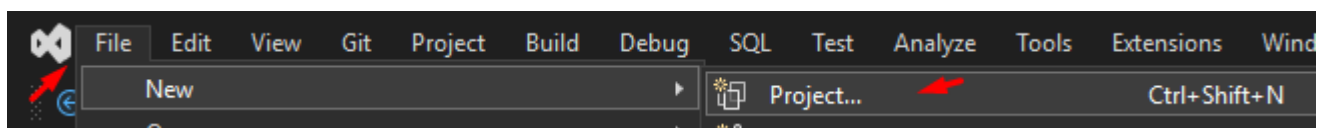


2. В столбец **Location** добавьте несколько городов.
3. Заметьте, что **Id** присваивается автоматически.

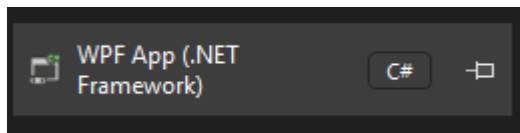


7. Создание WPF-проекта

1. В **Visual Studio** создайте новый проект: **File** → **New** → **Project**.



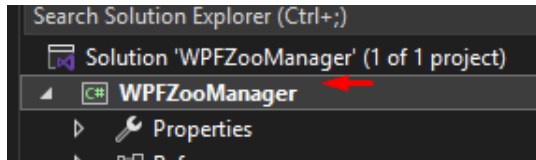
2. Выберите **WPF App (.NET Framework)** (обязательно этот тип, иначе база не подключится).



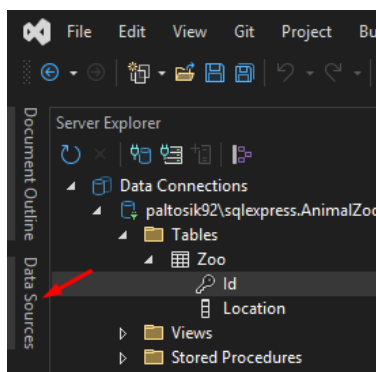
3. Назовите проект **WPFZooManager**.
4. Создайте **Git**-репозиторий для проекта.

8. Подключение базы данных

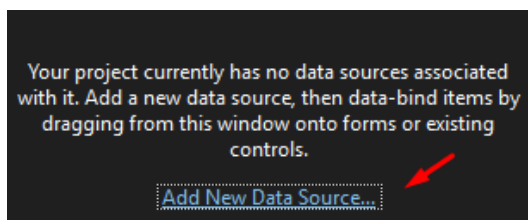
1. Выберите ваш проект в **Solution Explorer**.



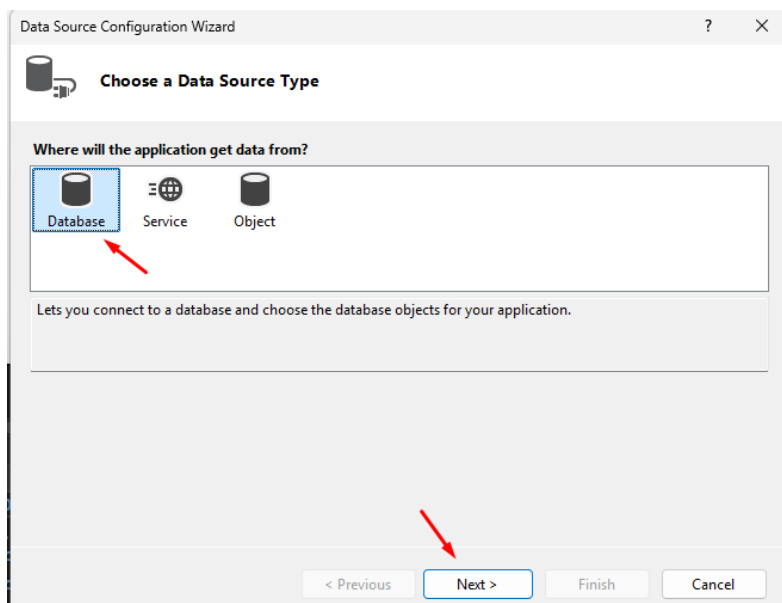
2. Откройте **View** → **Other Windows** → **Data Sources** (если нет — ищите в меню).

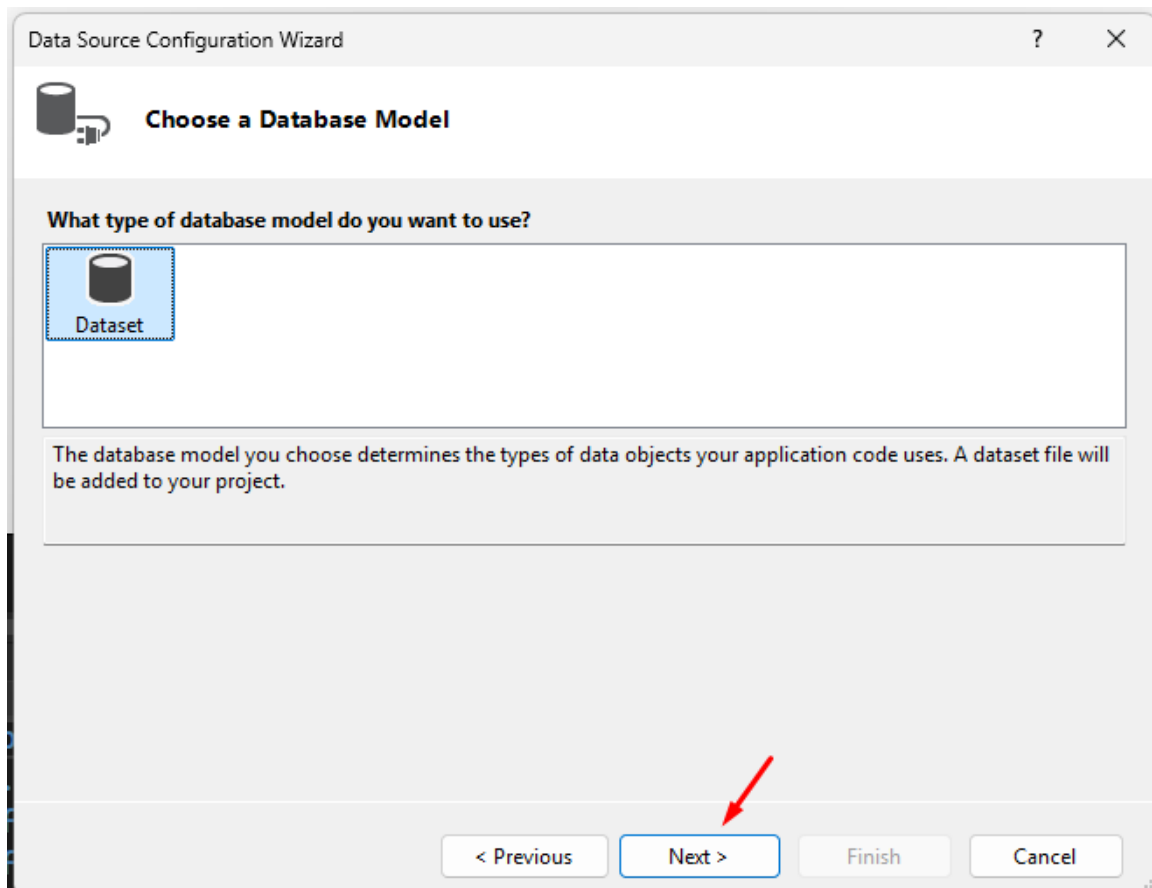


3. Нажмите **Add New Data Source....**

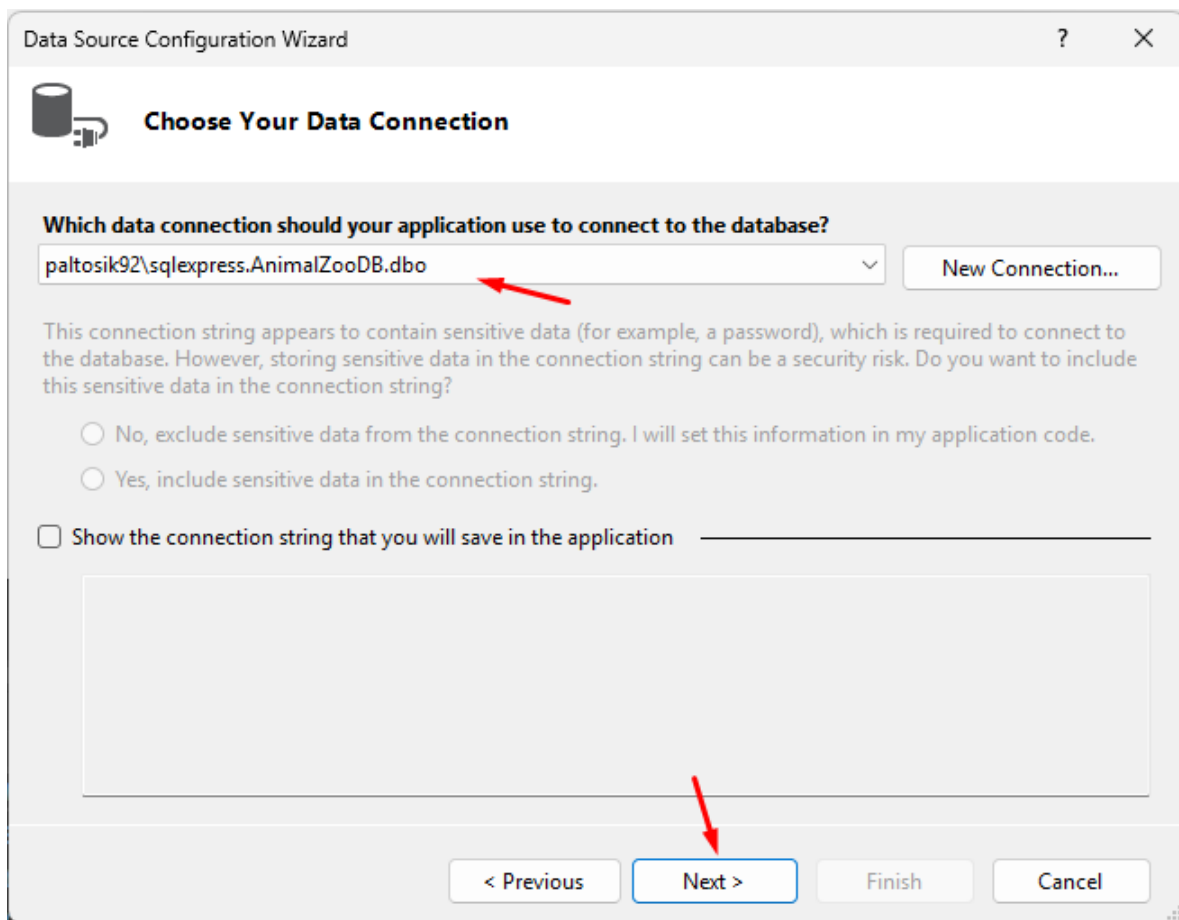


4. В мастере выберите **Database** → **Next** → **Next**.

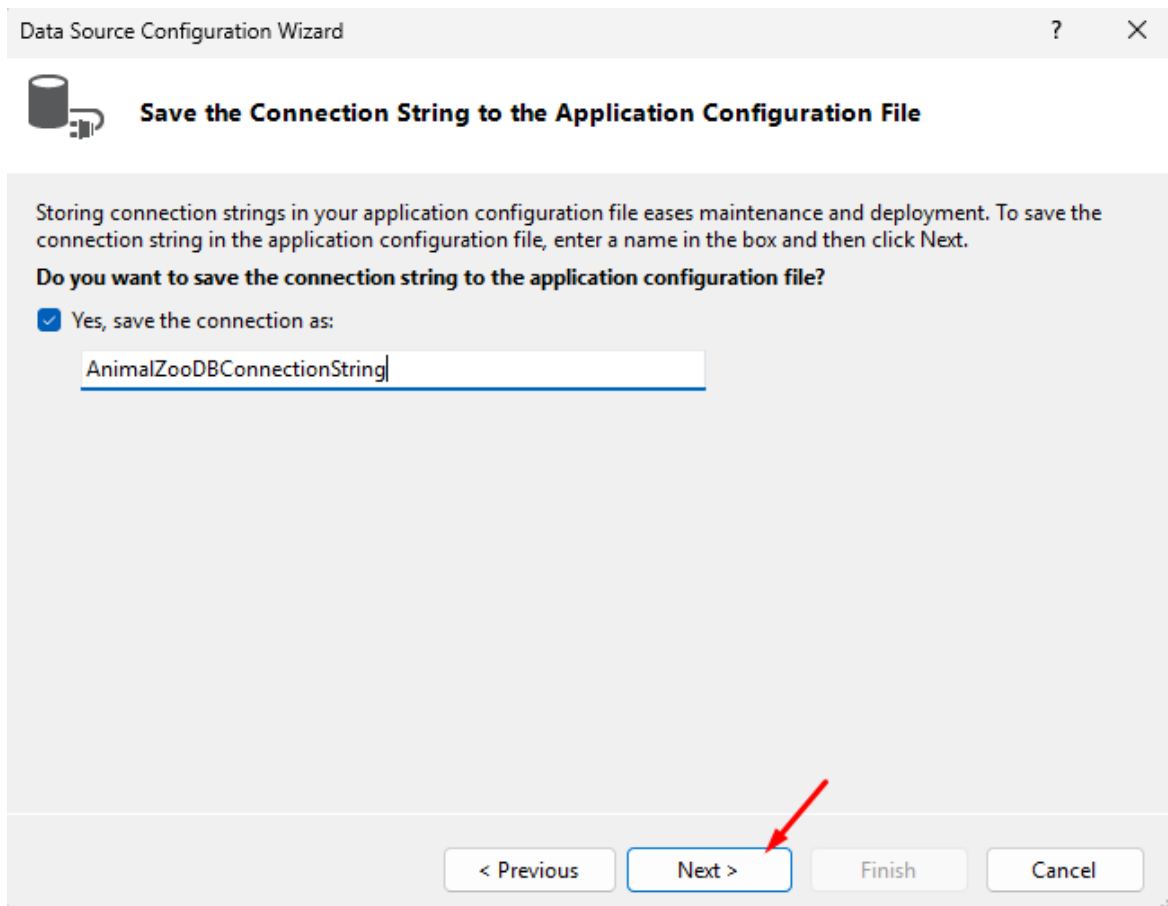




5. Выберите свою базу данных и нажмите **Next**.

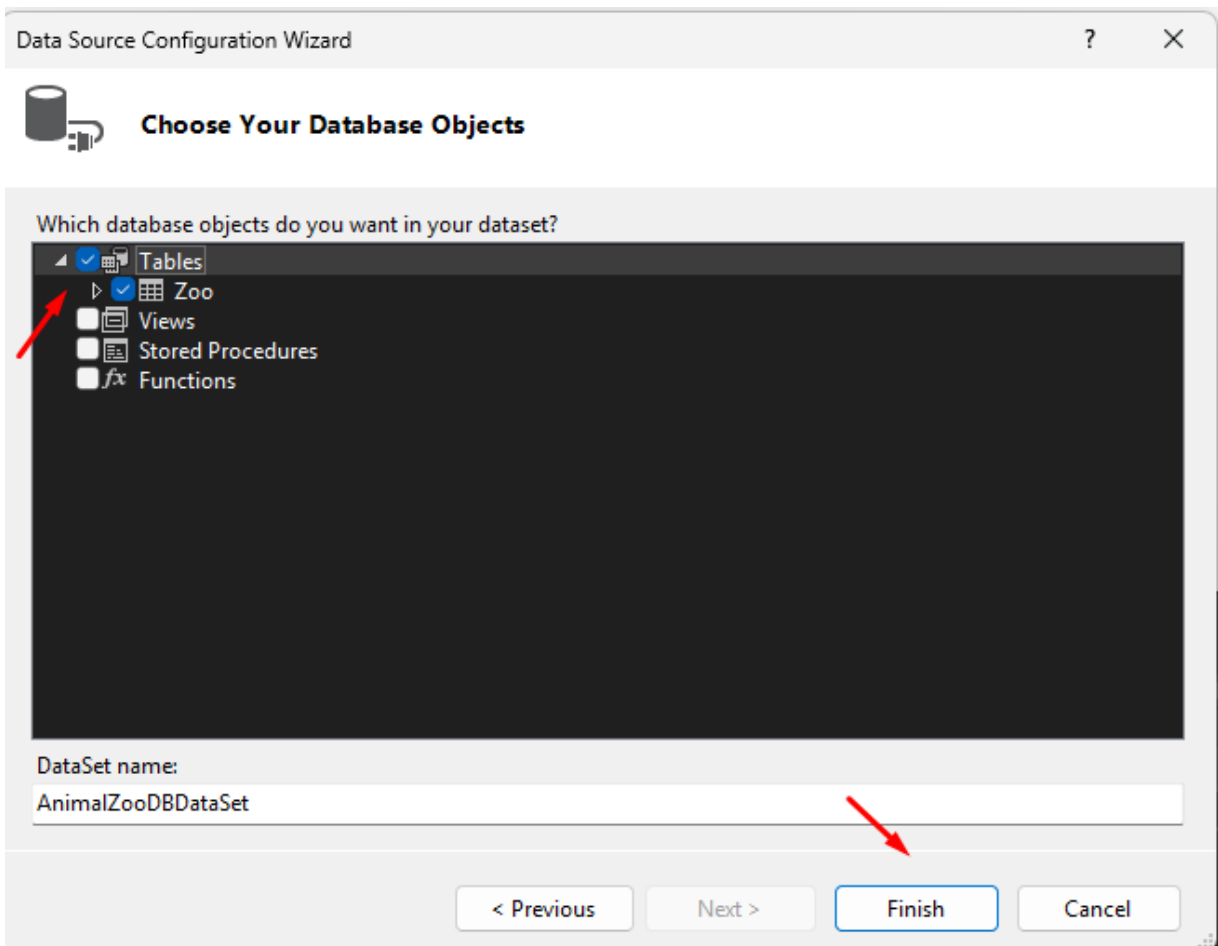


6. Запомните название строки подключения (**AnimalZooDBConnectionString**).



The screenshot shows the 'Data Source Configuration Wizard' window. The title bar says 'Data Source Configuration Wizard'. Below the title bar is a header with a database icon and the text 'Save the Connection String to the Application Configuration File'. The main area contains the following text: 'Storing connection strings in your application configuration file eases maintenance and deployment. To save the connection string in the application configuration file, enter a name in the box and then click Next.' Below this is a question: 'Do you want to save the connection string to the application configuration file?'. There is a checked checkbox labeled 'Yes, save the connection as:'. Below the checkbox is a text input field containing 'AnimalZooDBConnectionString'. At the bottom of the window are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. A red arrow points to the 'Next >' button.

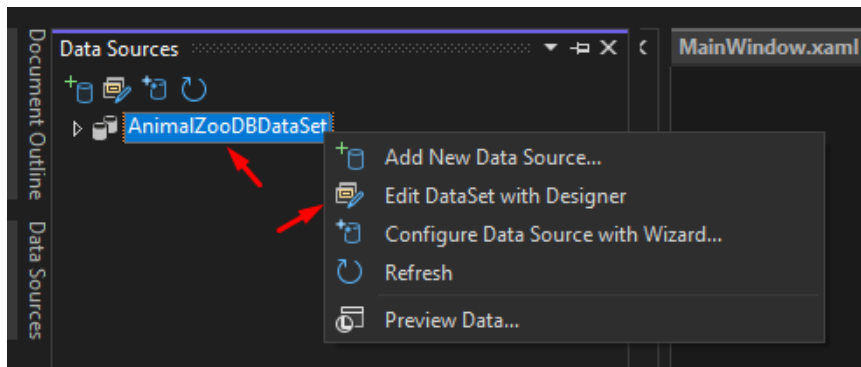
7. Поставьте галочки на нужных таблицах и нажмите **Finish**.



The screenshot shows the 'Data Source Configuration Wizard' window. The title bar says 'Data Source Configuration Wizard'. Below the title bar is a header with a database icon and the text 'Choose Your Database Objects'. The main area contains the following text: 'Which database objects do you want in your dataset?'. Below this is a tree view with the following items: 'Tables' (checked), 'Zoo' (checked), 'Views' (unchecked), 'Stored Procedures' (unchecked), and 'Functions' (unchecked). A red arrow points to the 'Tables' checkbox. Below the tree view is a text input field labeled 'DataSet name:' containing 'AnimalZooDBDataSet'. At the bottom of the window are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. A red arrow points to the 'Finish' button.

9. Редактирование DataSet

1. В окне **Data Sources** щёлкните ПКМ по своей таблице → **Edit DataSet with Designer**.



2. Это позволит настраивать отображение и связи между таблицами.

10. Подключение к базе данных в коде

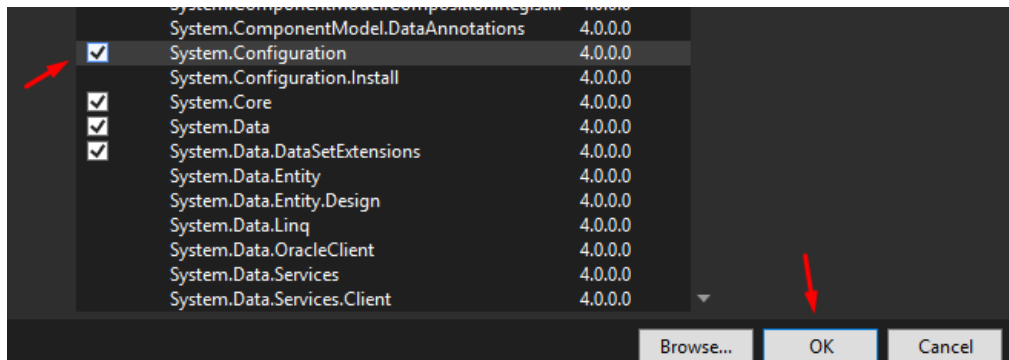
Теперь добавим строку подключения в коде:

1. Добавляем зависимость

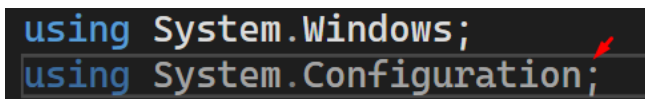
- В **Solution Explorer** щёлкните ПКМ по **References** → **Add Reference**.



- В списке найдите и отметьте **System.Configuration**.



2. Подключаем пространство имён



3. Получаем строку подключения в конструкторе окна

```
public MainWindow() {  
    InitializeComponent();  
    string connectionString = ConfigurationManager.ConnectionStrings  
        ["WPFZooManager.Properties.Settings.AnimalZooDBConnectionString"].ConnectionString;  
}
```

В `ConnectionStrings["..."]` указывается путь в формате:
<Имя проекта>.Properties.Settings.<Название_строки_подключения> — где

<Название_строки_подключения> совпадает с именем, которое вы запомнили при создании Data Source.

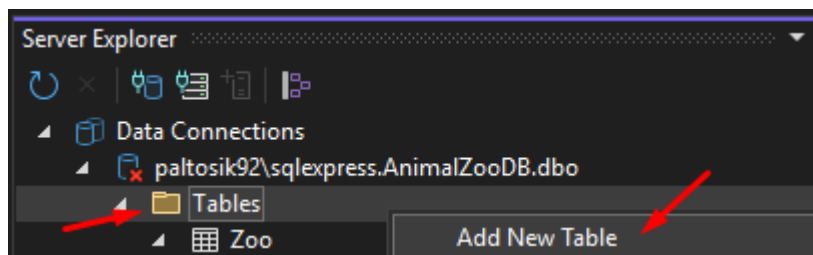
Шаг 8. Таблицы отношений или ассоциативные таблицы

В этом шаге мы создадим:

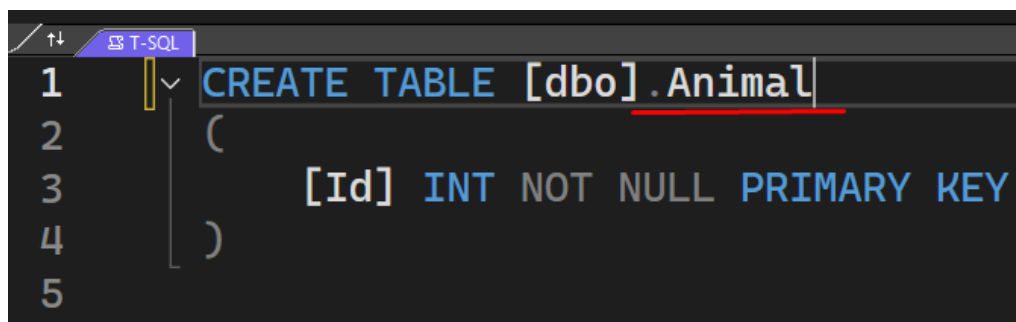
1. Таблицу **Animal** для хранения информации о животных.
2. Ассоциативную таблицу **ZooAnimal**, которая свяжет **Zoo** и **Animal** отношением «многие ко многим».

1. Создание таблицы Animal

1. В **Server Explorer** щёлкаем ПКМ по папке **Tables** и выбираем **Add New Table**:




2. В SQL вводим название **Animal**:

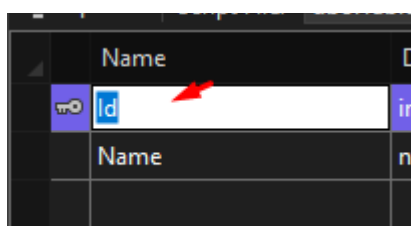


3. Создайте новый столбец:

- **Name** (тип nvarchar(50))
- **Data Type:** NVARCHAR(50)
- Снимите галочку **Allow Nulls**.

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	Name	nvarchar(50)	<input type="checkbox"/>	

4. Выберите столбец **Id** и нажмите **F4** для открытия окна **Properties**.




Найдите параметр **Identity Specification** → **Is Identity** и установите его в **True**.

Identity Specification	True
(Is Identity)	True
Identity Increment	1
Identity Seed	1


5. Внизу окна вы увидите SQL-запрос, соответствующий вашей таблице:

```
CREATE TABLE [dbo].[Animal]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name ] NVARCHAR(50) NOT NULL
)
```

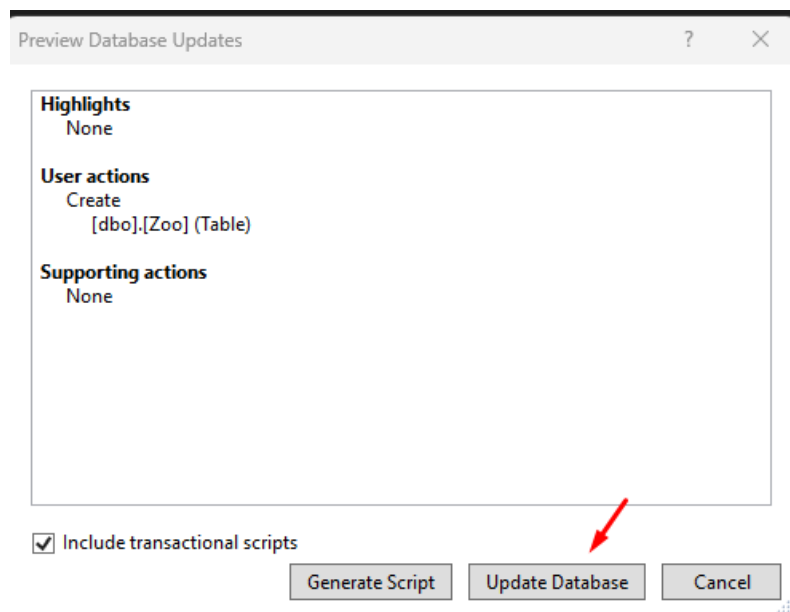
6. Нажмите кнопку **Update**:

 Update

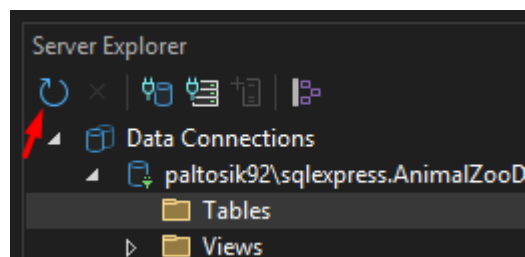
Script File: dbo.Table.sql*

Name	Data Type	Allow Nulls	Default
 Id	int	<input type="checkbox"/>	
Name	nvarchar(50)	<input type="checkbox"/>	
		<input type="checkbox"/>	

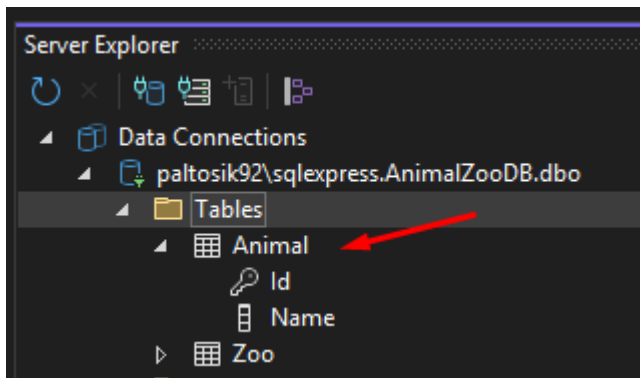
В появившемся окне выберите **Update Database**.



Нажмите **Refresh** в списке таблиц базы данных:

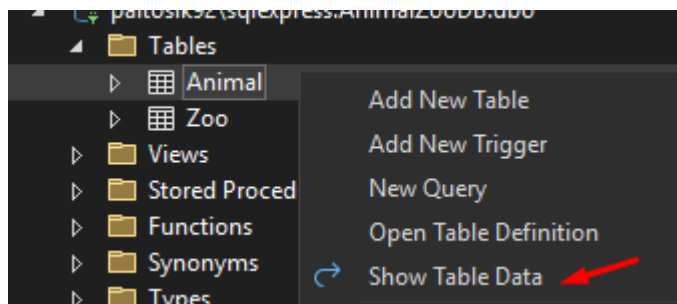


Убедитесь, что появилась таблица **Animal** со столбцами Id и Name.



2. Заполнение таблицы **Animal**

1. Щёлкните ПКМ по таблице **Animal** и выберите **Show Table Data**.



2. В столбец **Name** добавьте несколько названий - *Shark, Monkey, Wolf, Crocodile, Owl, Parrot*:

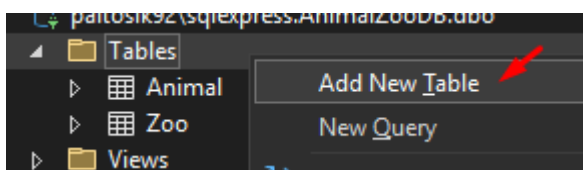
dbo.Animal [Data] dbo.Animal [Design]		
	Id	Name
	1	Shark
	2	Monkey
	3	Wolf
	4	Crocodile
	5	Owl
	6	Parrot
	NULL	NULL

Id будет проставляться автоматически.

3. Создание ассоциативной таблицы **ZooAnimal**

Теперь свяжем таблицу **Animal** с **Zoo**.

1. ПКМ по папке **Tables** → **Add New Table**.



2. Создайте столбцы:

- **Id** — int, Primary Key, автоинкремент.

Full Text Specification	False
Identity Specification	True
(Is Identity)	True

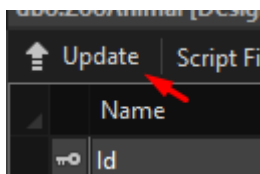
- **ZooID** — int, внешний ключ на Zoo(Id).
- **AnimalID** — int, внешний ключ на Animal(Id).

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	ZooID	int	<input type="checkbox"/>	
	AnimalID	int	<input type="checkbox"/>	
			<input type="checkbox"/>	

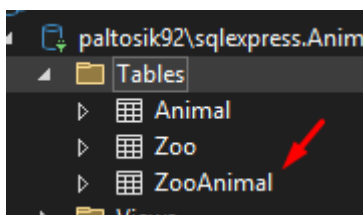
3. Переименуйте таблицу в **ZooAnimal**.

```
CREATE TABLE [dbo].[ZooAnimal]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [ZooID] INT NOT NULL,
    [AnimalID] INT NOT NULL
)
```

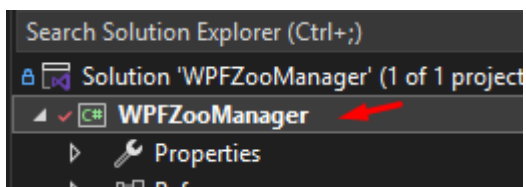
4. Сохраняем изменения (**Update Database**):



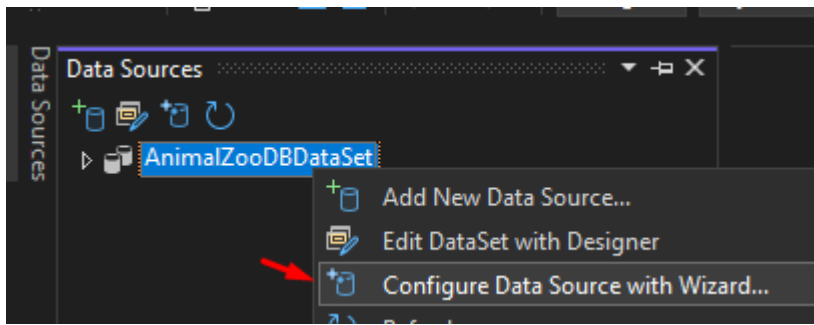
Обновляем и видим все три таблицы:



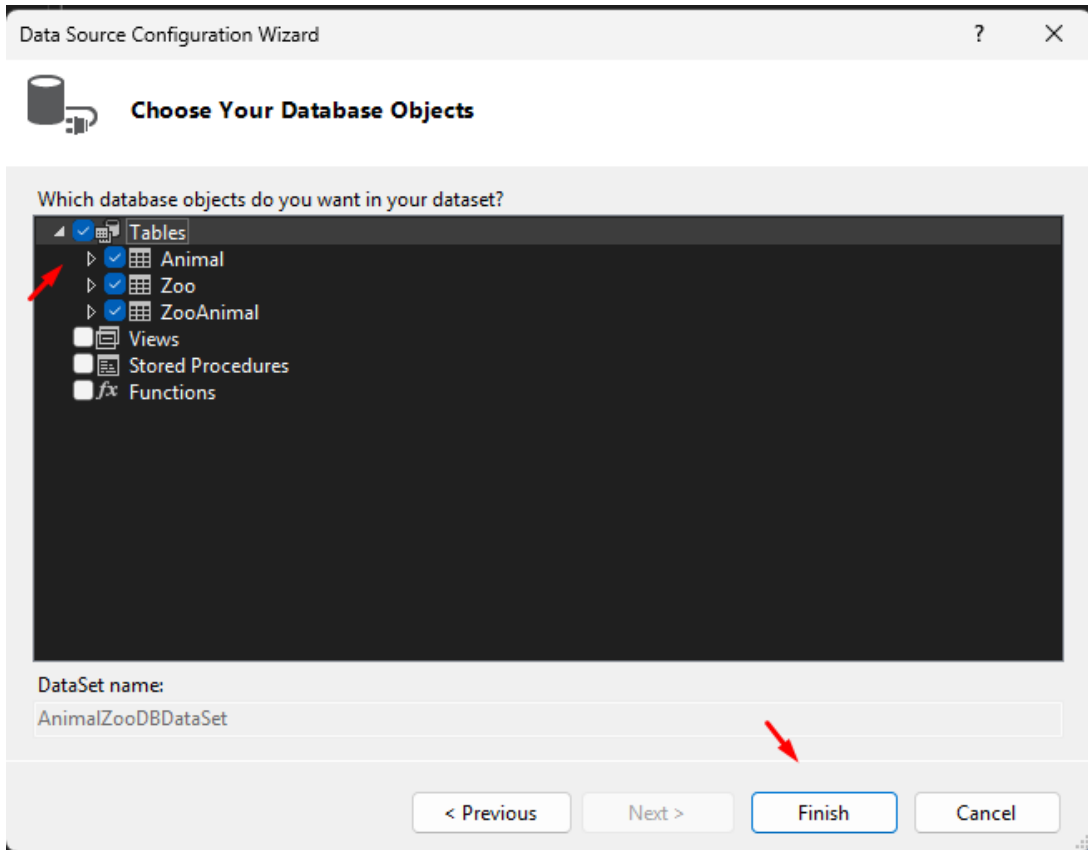
5. Далее выбираем наш проект:



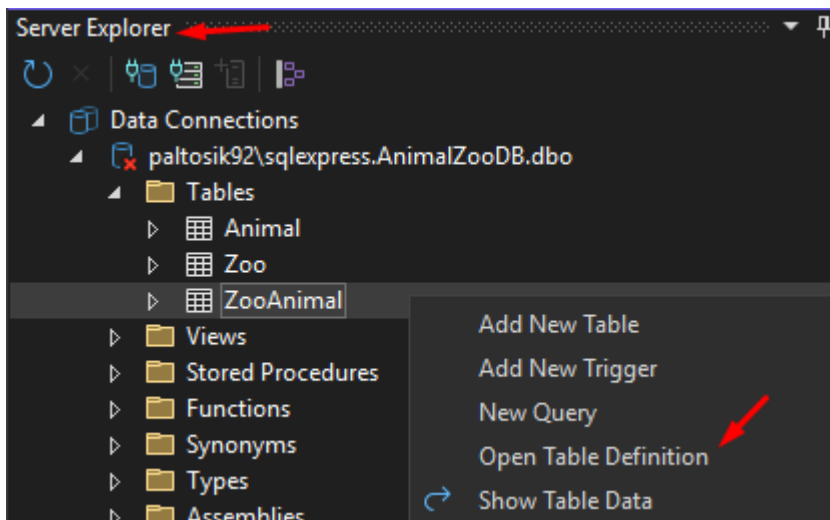
Открываем для него **Data Sources**, щёлкаем по нашей базе данных и выбираем **Configure Data Source with Wizard**:



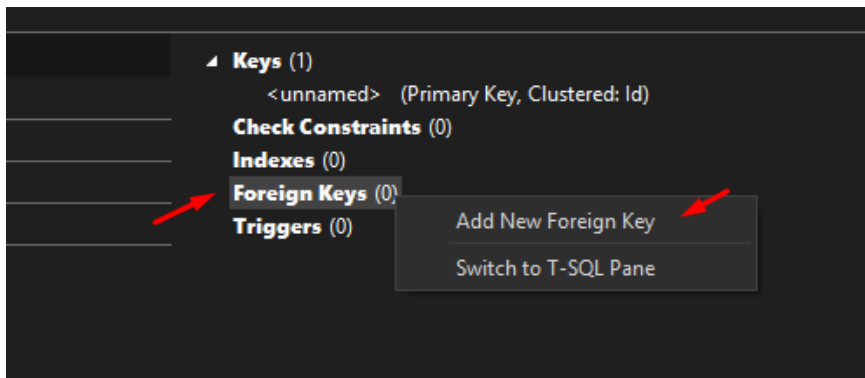
В окне ставим галочки у всех наших трёх таблиц и нажимаем **Finish**:



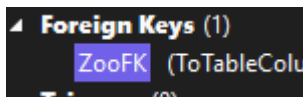
6. В Server Explorer открываем наши таблицы **ZooAnimal** (**Open Table Definition**):



7. Щёлкаем ПКМ по **Foreign Keys** и выбираем **Add New Foreign Key**:



Назовём его **ZooFK** (будет связывать ZooID с ID таблицы Zoo):



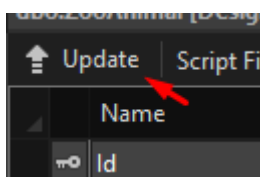
Обновим SQL добавив название баз и ключей, а также добавим новый ключ:

```
CREATE TABLE [dbo].[ZooAnimal]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [ZooID] INT NOT NULL,
    [AnimalID] INT NOT NULL,
    CONSTRAINT [ZooFK] FOREIGN KEY (ZooID) REFERENCES Zoo(Id),
    CONSTRAINT [AnimalFK] FOREIGN KEY (AnimalID) REFERENCES Animal(Id)
)
```

- CONSTRAINT [ZooFK] — задаёт имя ограничения для связи ZooID с Zoo(Id).
- CONSTRAINT [AnimalFK] — связь AnimalID с Animal(Id).

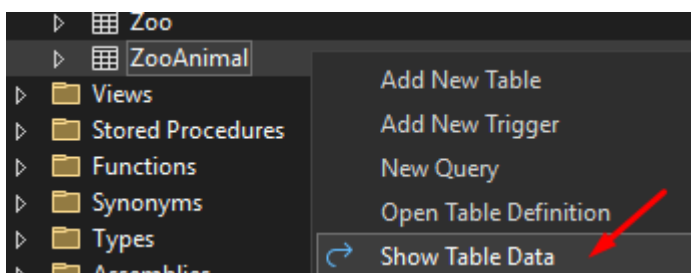
Это необходимо, чтобы база данных понимала, как таблицы связаны, и предотвращала некорректные данные.

8. Сохраняем изменения (**Update Database**):



5. Заполнение таблицы ZooAnimal

1. Откроем таблицу **ZooAnimal** (**Show Table Data**):



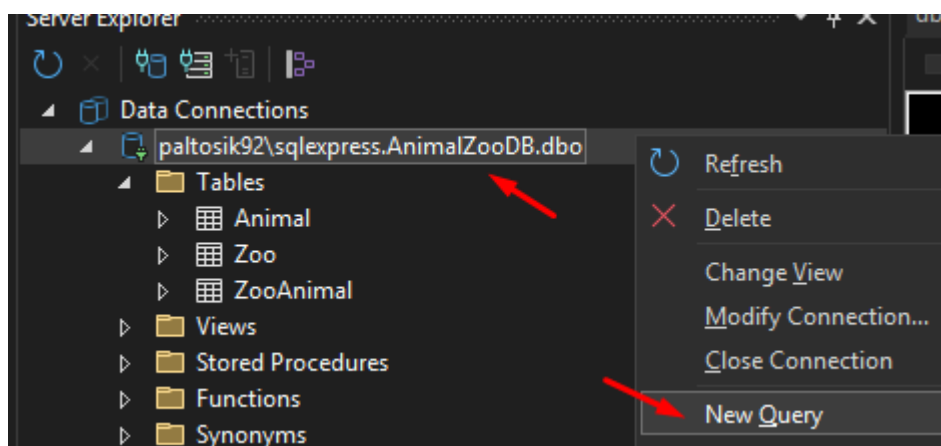
2. Заполним таблицу, добавив **ZooID** и **AnimalID**:

	Id	ZooID	AnimalID
	1	1	1
	3	1	2
	4	2	3
	5	3	5
	6	4	6
	8	4	3
	9	5	3
	NULL	NULL	NULL

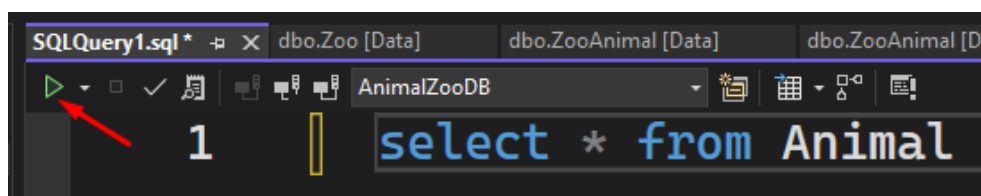
Как читать эти данные: Например, строка (1, 1, 1) означает, что в зоопарке с **Id=1** живёт животное с **Id=1**.

6. Примеры SQL-запросов

1. Мы можем проверить выборку наших животных. Щёлкаем **ПКМ** по базе данных и выбираем **New Query**:



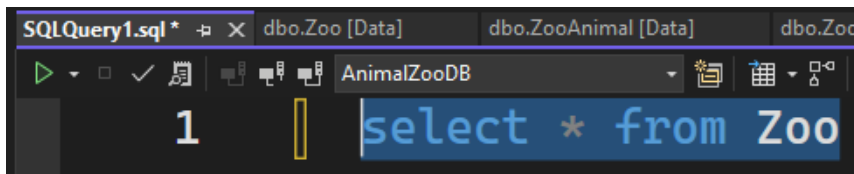
2. Вводим запрос и запускаем:



Выведет список всех животных с их Id:

	Id	Name
1	1	Shark
2	2	Monkey
3	3	Wolf
4	4	Crocodile
5	5	Owl
6	6	Parrot

2. Введём другой запрос для вывода всех зоопарков:



	Id	Location
1	1	Volgograd
2	2	Moscow
3	3	Tokyo
4	4	Shanghai
5	5	Paris

3. Животные в Волгограде:

А если мы хотим получить всех животных в Волгограде, то можем ввести:

```
select a.Name from Animal a inner join ZooAnimal
za on a.Id = za.AnimalID where za.ZooID = 1
```

Здесь мы:

- Соединяем таблицы **Animal** и **ZooAnimal** по полю AnimalID.
- Фильтруем по ZooID = 1 (Волгоград).

В результате получим: Shark, Monkey:

	Name
1	Shark
2	Monkey

Шаг 9. Отображение данных в списке

Перед тем как переходить к логике работы с данными, проверим корректность подключения базы данных и создадим простейший интерфейс для отображения информации.

1. Обновляем параметры окна

В файле **MainWindow.xaml** изменим размеры главного окна:

```
Title="MainWindow" Height="500" Width="725 ">
```

2. Добавляем элементы интерфейса

Внутри элемента **<Grid>** разместим **Label** и **ListBox**:

- Заголовок списка зоопарков

```
<Grid>
    <Label Content="Zoo List"
           HorizontalAlignment="Left"
           VerticalAlignment="Top"
           Margin="10,10,0,0"/>
</Grid>
```

- Список зоопарков

```
<ListBox x:Name="listZoos"
         HorizontalAlignment="Left"
         VerticalAlignment="Top"
         Width="155" Height="175"
         Margin="10,41,0,0"/>
</Grid>
```

Пояснение к элементам

- **Label** — отображает текст «Zoo List», служит заголовком для списка.
- **ListBox** (`x:Name="listZoos"`) — элемент для вывода списка зоопарков из базы данных.
- **HorizontalAlignment** / **VerticalAlignment** — выравнивание элемента в контейнере.
- **Margin** — отступы от границ контейнера (в пикселях).
- **Width** / **Height** — размеры элементов.

3. Подключаем пространства имён

В начале файла *MainWindow.xaml.cs* подключаем необходимые пространства имён:

```
using System.Windows;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;
```



```
SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, sqlConnection);
```

- **Назначение:** Создаёт мост между базой данных и приложением
- **Параметры:**
 - query - SQL-команда для выполнения
 - sqlConnection - объект подключения к БД
- **Особенности:**
 - Adapter автоматически управляет подключением (открывает/закрывает)
 - Не требует явного вызова Open()/Close()

- Объявляем блок using

```
using (sqlDataAdapter) {  
    |  
    |  
    |  
}
```

- **Назначение:** Гарантирует корректное освобождение ресурсов

- Создаём DataTable

```
using (sqlDataAdapter) {  
    |    DataTable zooTable = new DataTable();  
    |  
    |  
}
```

- **Назначение:** Создаёт таблицу в памяти для хранения результатов запроса
- **Аналогия:** Как лист Excel в оперативной памяти

- Заполняем DataTable

```
using (sqlDataAdapter) {  
    |    DataTable zooTable = new DataTable();  
    |    sqlDataAdapter.Fill(zooTable);  
    |  
}
```

- **Что происходит:**
 1. Adapter открывает соединение
 2. Выполняет SQL-запрос
 3. Получает результаты
 4. Заполняет ими zooTable
 5. Закрывает соединение

- Настраиваем ListBox


```
using (sqlDataAdapter) {
    DataTable zooTable = new DataTable();
    sqlDataAdapter.Fill(zooTable);
    listZoos.DisplayMemberPath = "Location";
    listZoos.SelectedValuePath = "Id";
    listZoos.ItemsSource = zooTable.DefaultView;
}
```

- **Назначение DisplayMemberPath:** Указывает, какое поле из DataTable отображать в списке
- **Назначение SelectedValuePath:** Указывает, какое поле использовать как скрытое значение
- **Назначение ItemsSource:** Устанавливает источник данных для ListBox

Полный процесс работы метода

1. Формируется SQL-запрос
2. Создаётся адаптер для выполнения запроса
3. Создаётся таблица в памяти
4. Адаптер заполняет таблицу данными из БД
5. Настраивается привязка данных к ListBox:
 - Что показывать (Location)
 - Что хранить (Id)
 - Откуда брать данные (zooTable.DefaultView)

Полный код метода:

```
private void ShowZoos() {
    string query = "select * from Zoo";
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter
        (query, sqlConnection);
    using (sqlDataAdapter) {
        DataTable zooTable = new DataTable();
        sqlDataAdapter.Fill(zooTable);
        listZoos.DisplayMemberPath = "Location";
        listZoos.SelectedValuePath = "Id";
        listZoos.ItemsSource = zooTable.DefaultView;
    }
}
```

6. Не забываем вызывать метод в **MainWindow()**:

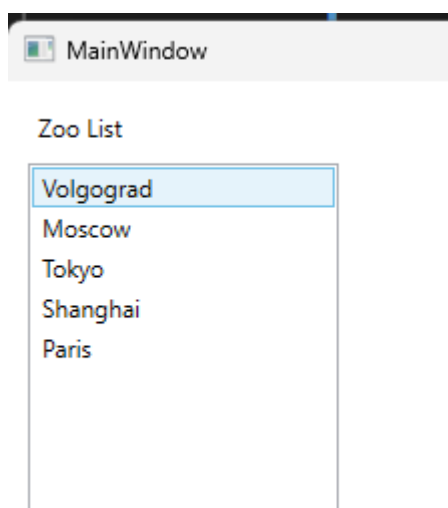
```

public partial class MainWindow : Window {
    SqlConnection sqlConnection;
    public MainWindow() {
        InitializeComponent();
        string connectionString = Configurati
            ["WPFZooManager.Properties.Settings
            nString;
        sqlConnection = new SqlConnection(co
        ShowZoos();
    }
}

```

7. Результат

После запуска приложения в ListBox будет отображён список городов, хранящихся в таблице Zoo.Id каждого города хранится «за кадром» и может использоваться для дальнейших запросов.



8. Более правильно будет обернуть нашу конструкцию в **try/cath** на случай возникновения ошибок:

```

private void ShowZoos() {
    try {
        string query = "select * from Zoo";
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, sqlConnection);

        using (sqlDataAdapter) {
            DataTable zooTable = new DataTable();
            sqlDataAdapter.Fill(zooTable);
            listZoos.DisplayMemberPath = "Location";
            listZoos.SelectedValuePath = "Id";
            listZoos.ItemsSource = zooTable.DefaultView;
        }
    } catch (Exception e) {
        MessageBox.Show(e.ToString());
    }
}

```

Шаг 10. Список животных, привязанных к выбранному зоопарку

В этом шаге мы добавим второй список, в котором будут отображаться животные, находящиеся в выбранном зоопарке.

У нас уже есть:

- список зоопарков (**ListBox listZoos**),
- таблица животных в базе данных.

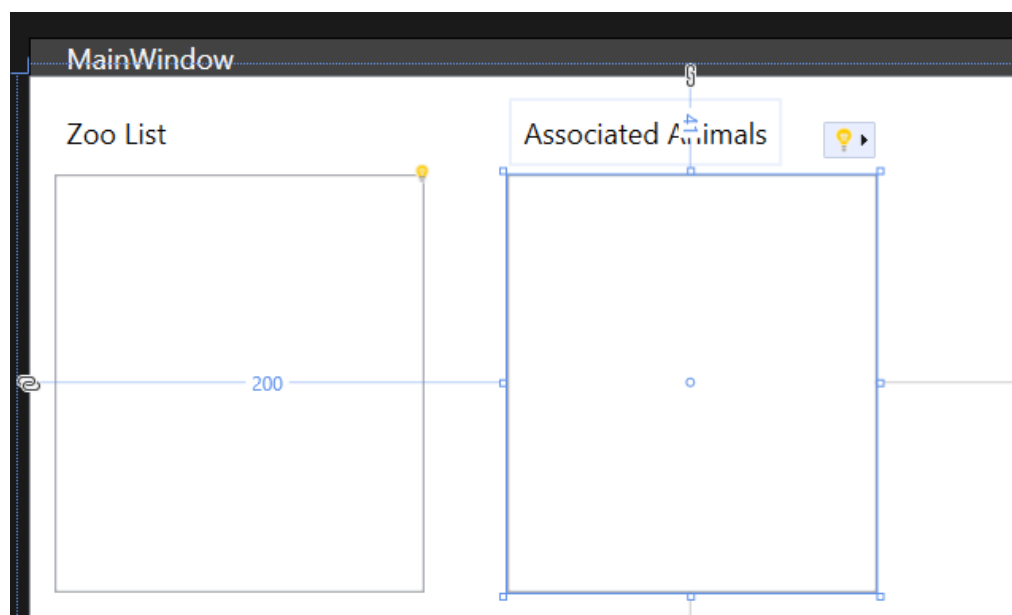
Теперь мы хотим, чтобы при выборе зоопарка в отдельном ListBox отображались только те животные, которые относятся именно к нему.

1. Изменение XAML

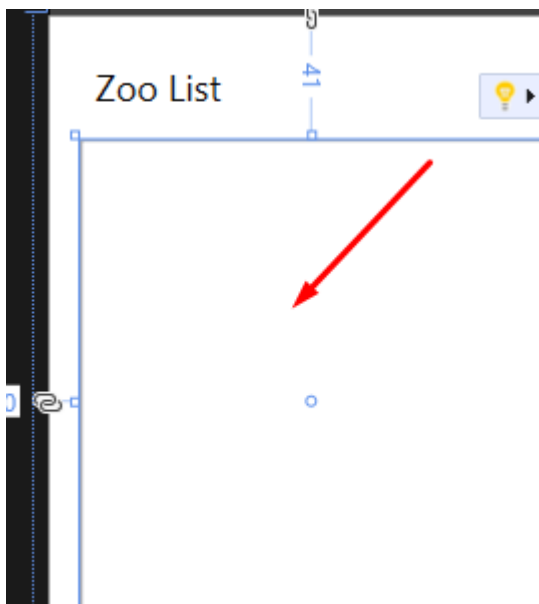
В файле **MainWindow.xaml** добавим ещё один ListBox для животных. Можно скопировать уже существующий блок и изменить имя, надпись и расположение:

```
<!-- Список животных, связанных с зоопарком -->
<Label Content="Associated Animals"
      HorizontalAlignment="Left"
      VerticalAlignment="Top"
      Margin="202,10,0,0"/>
<ListBox x:Name="listAssociatedAnimals"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="155" Height="175"
        Margin="200,41,0,0"/>
</Grid>
```

Итог:



Дважды щёлкните по ListBox:



В итоге он добавится в качестве метода в код, который будет выполняться при выборе зоопарка:

```
private void listZoos_SelectionChanged(object sender,
    System.Windows.Controls.SelectionChangedEventArgs e) {
}

```

А в Xaml размете появится новое свойство:

```
<ListBox x:Name="listZoos"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Width="155" Height="175"
    Margin="10,41,0,0"
    SelectionChanged="listZoos_SelectionChanged"/>

```

2. Создание метода ShowAssociatedAnimals()

Метод будет похож на ShowZoos(), но с SQL-запросом, который соединяет таблицы Animal, ZooAnimal и Zoo. Поэтому можете скопировать ShowZoos(), и обновить код:

```
private void ShowAssociatedAnimals() {
    try {
        string query = @"SELECT a.Name FROM Animal a
                        INNER JOIN ZooAnimal za ON a.Id = za.AnimalId
                        WHERE za.ZooId = @ZooId";

        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand);

        using (sqlDataAdapter) {
            DataTable animalTable = new DataTable();
            sqlDataAdapter.Fill(animalTable);

            listAssociatedAnimals.DisplayMemberPath = "Name";
            listAssociatedAnimals.SelectedValuePath = "Id";
            listAssociatedAnimals.ItemsSource = animalTable.DefaultView;
        }
    } catch (Exception e) {
        MessageBox.Show(e.ToString());
    }
}
```

И не забудьте вызывать его в методе `listZoos_SelectionChanged()`:

```
private void listZoos_SelectionChanged(object sender,
    System.Windows.Controls.SelectionChangedEventArgs e) {
    ShowAssociatedAnimals();
}
```

Как это работает

1. При выборе зоопарка вызывается `listZoos_SelectionChanged`.
2. В методе `ShowAssociatedAnimals()` выполняется запрос с параметром `@ZooId`.
3. `@ZooId` получает значение из `listZoos.SelectedValue` (ID зоопарка).
4. В результате `ListBox listAssociatedAnimals` показывает список животных, привязанных к выбранному зоопарку.

Запустите программу и проверьте работу:

Zoo List	Associated Animals
<div>Volgograd</div> <div>Moscow</div> <div>Tokyo</div> <div>Shanghai</div> <div>Paris</div>	<div>Shark</div> <div>Monkey</div>

Шаг 11. Выводим список всех животных из базы

Теперь, когда у нас есть таблица с зоопарками, и таблица с животными, привязанными к конкретному зоопарку, мы добавим третий список — **полный список всех животных** из базы данных.

Задача: вывести в новый ListBox все записи из таблицы Animal.

1. Добавляем элемент в XAML

Сначала в файле. xaml копируем уже существующий ListBox (например, тот, что для связанных животных) и вставляем его рядом. Переименовываем новый элемент в:

x:Name="listAllAnimals", меняем размеры и положение:

```
<!-- Список всех животных -->
<ListBox x:Name="listAllAnimals"
          HorizontalAlignment="Left"
          VerticalAlignment="Top"
          Width="155" Height="300"
          Margin="396,41,0,0"/>
</Grid>
```

2. Создаём метод ShowAllAnimals

В файле *MainWindow.xaml.cs* пишем новый метод:

```
private void ShowAllAnimals() {
    try {
        string query = "SELECT * FROM Animal";
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, sqlConnection);

        using (sqlDataAdapter) {
            DataTable animalTable = new DataTable();
            sqlDataAdapter.Fill(animalTable);

            listAllAnimals.DisplayMemberPath = "Name";
            listAllAnimals.SelectedValuePath = "Id";
            listAllAnimals.ItemsSource = animalTable.DefaultView;
        }
    } catch (Exception e) {
        MessageBox.Show(e.ToString());
    }
}
```

3. Вызываем метод при старте программы

В конструкторе MainWindow после инициализации добавляем:

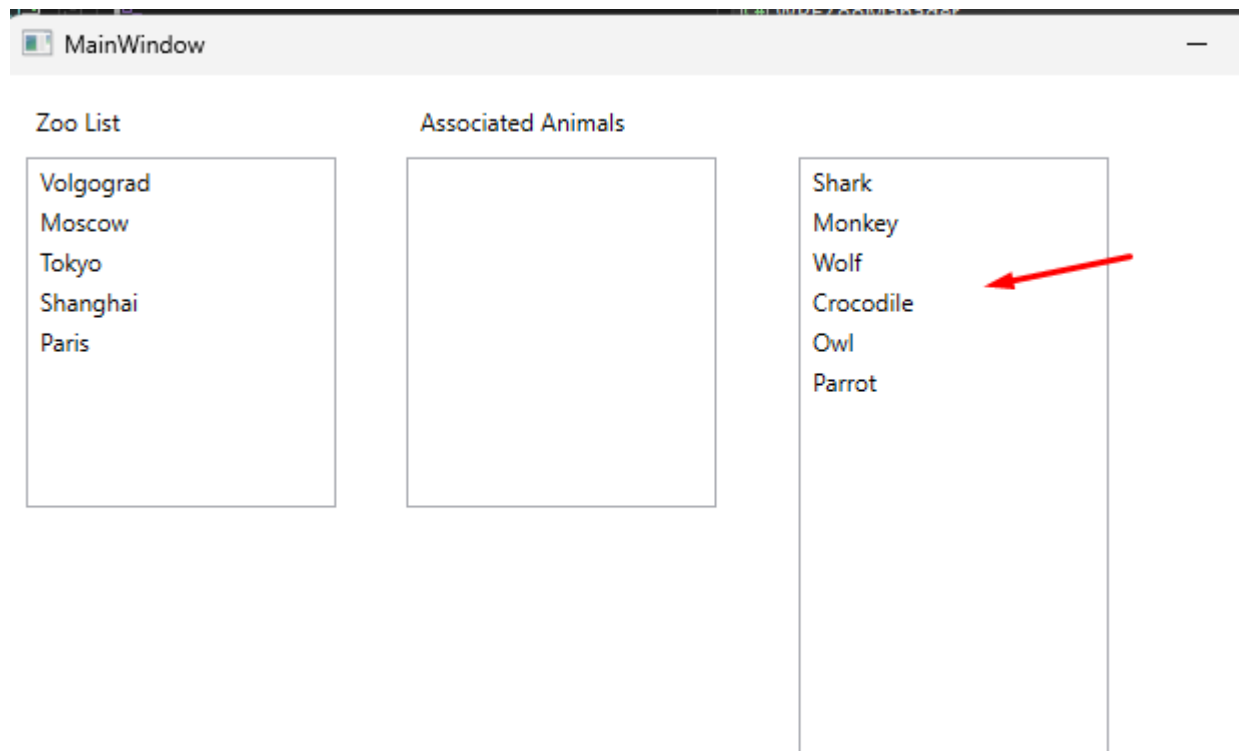
```

public partial class MainWindow : Window {
    SqlConnection sqlConnection;
    public MainWindow() {
        InitializeComponent();
        string connectionString = Configurati
        ["WPFZooManager.Properties.Setting
        sqlConnection = new SqlConnection(co
        ShowZoos();
        → ShowAllAnimals();
    }
}

```

4. Результат

После запуска вы увидите всех животных:



Шаг 12. Доделываем дизайн нашего приложения

Приведём наше приложение к финальному виду и в следующих шагах реализуем функциональность элементов.

1. В начале изменим ширину и высоту:

```

Title="MainWindow" Height="450" Width="580">

```

2. Затем добавим кнопки в наше приложение:


```
<Button Content="Delete Zoo"
        Click="DeleteZoo_Click"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="10,235,0,0"/>
```

```
<Button Content="Remove Animal"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="202,235,0,0"/>
```

```
<Button Content="Add Zoo"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="10,318,0,0" />
```

```
<Button Content="Add Animal Zoo"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="202,318,0,0"/>
```

```
<Button Content="Update Zoo"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="10,357,0,0"/>
```

```
<Button Content="Update Animal"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="202,357,0,0"/>
```

```
<Button Content="Delete Animal"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="396,357,0,0"/>
```

```
<Button Content="Add Animal to Zoo"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="158" Height="34"
        Margin="396,6,0,0"/>
```

3. А также TextBox:

```
<TextBox Text="TextBox"
        TextWrapping="Wrap"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Height="39" Width="350"
        Margin="10,274,0,0"/>
```

Шаг 13. Реализуем кнопку удаления

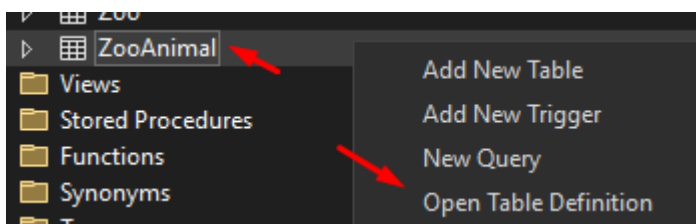
1. Теперь реализуем нашу кнопку в коде:

```
private void DeleteZoo_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "delete from Zoo where id = @ZooId";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);
        sqlCommand.ExecuteScalar();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowZoos();
    }
}
```


Разбор:

- `string query = "delete from Zoo where id = @ZooId";` — SQL-запрос удаляет запись из таблицы Zoo по конкретному Id.
- `SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);` — создаёт команду, привязанную к соединению.
- `sqlConnection.Open();` — открывает подключение к базе.
- `sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);` — подставляет в запрос выбранное значение из ListBox (ID зоопарка).
- `sqlCommand.ExecuteScalar();` — выполняет запрос.
- `finally` — в любом случае закрываем соединение и обновляем список зоопарков (`ShowZoos()`).

5. Далее нам нужно обновить **SQL** запрос у базы данных **ZooAnimal** (ПКМ по базе и выбираем **Open Table Definition**):



Добавляем в последние строки **ON DELETE CASCADE**:

```
CREATE TABLE [dbo].[ZooAnimal]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [ZooID] INT NOT NULL,
    [AnimalID] INT NOT NULL,
    CONSTRAINT [ZooFK] FOREIGN KEY (ZooID) REFERENCES Zoo(Id) ON DELETE CASCADE,
    CONSTRAINT [AnimalFK] FOREIGN KEY (AnimalID) REFERENCES Animal(Id) ON DELETE CASCADE
)
```

И обновляем базу через **Update**.

Что делает ON DELETE CASCADE:

Это правило для внешнего ключа, которое говорит:

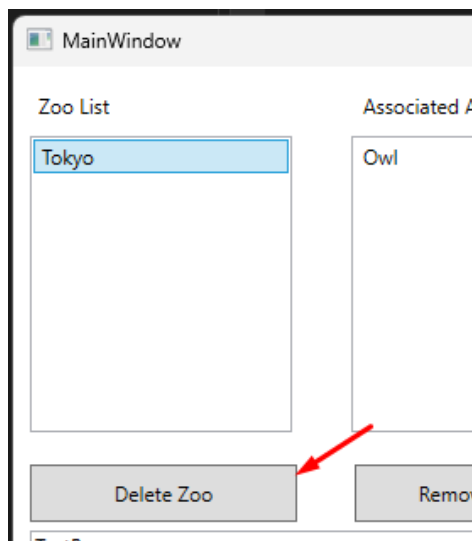
Если в родительской таблице (Zoo или Animal) удаляется запись, то все связанные строки в таблице ZooAnimal удаляются автоматически.

Пример:

- У вас есть Zoo с Id = 5.
- В ZooAnimal есть 5 животных, привязанных к этому зоопарку.

- Если удалить зоопарк Id = 5, то **все 5 строк** с этим ZooID из ZooAnimal удалятся **автоматически**.
- Не надо писать отдельный DELETE для ZooAnimal.

Теперь запустим приложение и проверим работу кнопки:



Вы сразу заметите, что у нас появляется исключение. Это связано с тем, что при удалении зоопарка у него есть связанные животные.

Можем временно убрать его, закомментировав у метода `ShowAssociatedAnimals()`:

```
    } catch (Exception e) {
        //MessageBox.Show(e.ToString());
    }
```

Теперь с каскадом ошибки не будет, и этот MessageBox временно не нужен для проверки.

Шаг 14. Реализация кнопок

В `TextBox` добавляем имя:

```
<TextBox x:Name="myTextBox"
```

Для кнопки **Add Zoo** указываем обработчик:

```
<Button Content="Add Zoo"
        Click="AddZoo_Click"
```

Добавляем в код обработчик событий:

```
private void AddZoo_Click(object sender, RoutedEventArgs e) {
    // ...
}
```

Скопируем код из кнопки DeleteZoo внутрь:

```
private void AddZoo_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "delete from Zoo where id = @ZooId";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);
        sqlCommand.ExecuteNonQuery();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowZoos();
    }
}
```

И внесём изменения:

```
string query = "insert into Zoo values (@Location)";
```

SQL-запрос, добавляющий новую запись в таблицу Zoo.

```
sqlCommand.Parameters.AddWithValue("@Location", myTextBox.Text);
```

@Location — параметр, который мы задаём через myTextBox.Text.

После выполнения запроса вызываем ShowZoos(), чтобы список обновился.

Запускаем, и проверяем что у нас теперь добавляются города:

Zoo List

Для кнопки **Add Animal to Zoo** указываем клик:

```
<Button Content="Add Animal to Zoo"
        Click="AddAnimalToZoo_Click"
```

Копируем код из любой кнопки и вносим правки:

```
string query = "insert into ZooAnimal values (@ZooId, @AnimalId)";
```

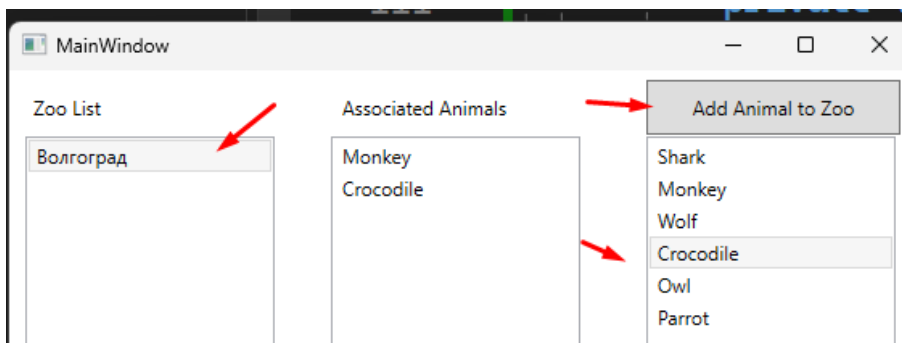
```
sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);  
sqlCommand.Parameters.AddWithValue("@AnimalId", listAllAnimals.SelectedValue);
```

Таблица ZooAnimal связывает Zoo и Animal. В неё добавляется пара ZooId (выбранный зоопарк) и AnimalId (выбранное животное).

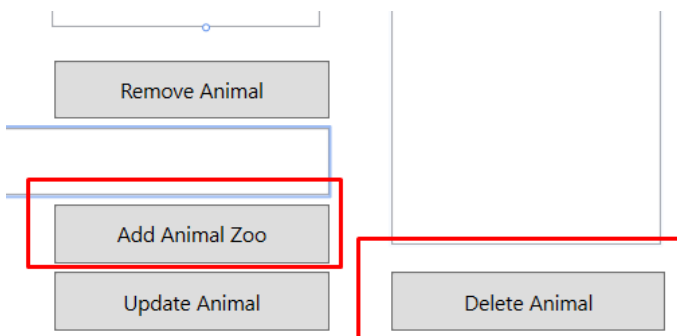
В блоке finally после вставки обновляем список животных в текущем зоопарке (ShowAssociatedAnimals()):

```
} finally {  
    sqlConnection.Close();  
    ShowAssociatedAnimals();  
}
```

Запускаем и проверяем работу кнопки:



Реализуем кнопки **Add Animal** и **Delete Animal**, чтобы управлять таблицей Animal.



Обновляем XAML

Для кнопки добавления животного:

```
<Button Content="Add Animal Zoo"  
        Click="AddAnimal_Click"/>
```

Для кнопки удаления животного:

```
<Button Content="Delete Animal"  
        Click="DeleteAnimal_Click"/>
```

Реализуем кнопку добавления нового животного:

```
private void AddAnimal_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "insert into Animal values (@Name)";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@Name", myTextBox.Text);
        sqlCommand.ExecuteNonQuery();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowAllAnimals();
    }
}
```

- Добавляем новую запись в таблицу Animal.
- Имя берётся из myTextBox.Text.
- После вставки вызываем ShowAllAnimals(), чтобы сразу увидеть результат.

Реализуем кнопку удаления выбранного животного:

```
private void DeleteAnimal_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "delete from Animal where id = @AnimalId";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@AnimalId", listAllAnimals.SelectedValue);
        sqlCommand.ExecuteNonQuery();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowAllAnimals();
    }
}
```

- SQL-запрос удаляет запись по выбранному AnimalId.
- listAllAnimals.SelectedValue — это ID текущего животного в списке.
- После удаления обновляем список.

Запускаем приложение и проверяем работу:

1. Вводим название животного в TextBox → жмём **Add Animal**.
→ животное появляется в списке.
2. Выбираем любое животное → жмём **Delete Animal**.
→ животное исчезает из списка и из БД.

Добавляем для остальных кнопок клики:

```
<Button Content="Remove Animal"
        Click="RemoveAnimal_Click"/>
```

```
<Button Content="Update Zoo"
        Click="UpdateZoo_Click"/>
```

```
<Button Content="Update Animal"
        Click="UpdateAnimal_Click"/>
```

Реализуем для них код:

```
private void UpdateZoo_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "update Zoo Set Location = @Location where Id = @ZooId";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@ZooId", listZoos.SelectedValue);
        sqlCommand.Parameters.AddWithValue("@Location", myTextBox.Text);
        sqlCommand.ExecuteNonQuery();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowZoos();
    }
}
```

```
private void UpdateAnimal_Click(object sender, RoutedEventArgs e) {
    try {
        string query = "update Animal Set Name = @Name where Id = @AnimalId";
        SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);
        sqlConnection.Open();
        sqlCommand.Parameters.AddWithValue("@AnimalId", listAllAnimals.SelectedValue);
        sqlCommand.Parameters.AddWithValue("@Name", myTextBox.Text);
        sqlCommand.ExecuteNonQuery();
    } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
    } finally {
        sqlConnection.Close();
        ShowAllAnimals();
    }
}
```

Самостоятельные задания

Оформить проект **ZooManager** и выложить его на GitHub.

Требования к выполнению:

1. Визуальное оформление

- Разместите все кнопки и списки аккуратно в окне приложения.
- Используйте **Grid** или **StackPanel** для структуры.
- Добавьте отступы (Margin, Padding) и сделайте кнопки одинакового размера.
- Примените базовое цветовое оформление (например, разные цвета для кнопок *Add*, *Update*, *Delete*).

2. Функциональность

- Проверьте, что все реализованные кнопки работают:
 - Add Zoo
 - Add Animal
 - Delete Zoo
 - Delete Animal
 - Add Animal to Zoo
 - Remove Animal from Zoo
 - Update Zoo
 - Update Animal

3. GitHub

- Создайте публичный репозиторий с названием **ZooManager**.

- Загрузите туда весь проект.
- В README.md опишите:
 - назначение программы,
 - какие функции реализованы,
 - как запустить проект.

4. Сдача работы

- Отправьте ссылку на свой репозиторий GitHub с рабочим приложением.