



Advanced Programming C#

Lecture 5

dr inż. Małgorzata Janik
malgorzata.janik@pw.edu.pl



Today you will need:



Magazynowanie i przetwarzanie danych



Łączenie, programowanie i testowanie rozwiązań do obsługi danych przy użyciu programu SQL Server, usługi...

✓ Magazynowanie i przetwarzanie danych

Opcjonalnie

- ☒ SQL Server Data Tools
- ☒ Narzędzia usługi Azure Data Lake i Stream Analy...
- ☒ Narzędzia programistyczne programu .NET Fram...
- ☒ Wyszukiwanie Redgate SQL
- ☐ Obsługa języka F# dla komputerów

Visual Studio Installer

Installed Available

All installations are up to date.



Visual Studio Community 2022

17.7.6

Powerful IDE, free for students, open-source contributors, and individuals

[Release notes](#)

Modify

Launch

More ▼



PROJECTS!

Today I would like to
receive proposals for the
Projects. If not done please
send 1-2 sentences via
Teams.

Project I

- Few summary slides about the projects.
- Presentation must include:
 - Idea, description & specification of the project
 - used technologies
 - Screenshots of the prototype of the application
- Presentation must be put in the github and then the link sent to malgorzata.janik@pw.edu.pl

Deadline: 2 weeks (1.12.25)



ADO .NET (Databases)



Relational databases SQL (reminder)

Databases (REMINDER)

- Relational databases
 - Data is grouped in tables

<u>Id</u>	name	surname	birth date
1	Małgorzata	Janik	1887
2	Helena	Nowak	1984

- SQL - Structured Query Language is a programming language used to manage data in relational databases.



Databases (REMINDER)

- Basic SQL queries:
 - **SELECT** - used to read (or select) your data
 - SELECT* from TableName
 - SELECT Column1, Column2 from TableName
 - **INSERT INTO** - used when you want to add (or insert) new data
 - INSERT INTO TableName VALUES (Value1, Value2);
 - INSERT INTO TableName(Column2,Column3) VALUES (Value1, Value2);
 - **DELETE** - used to remove row(s)
 - DELETE FROM table_name WHERE *condition*;
 - DELETE FROM OSOBY WHERE imie = 'Malgorzata';



Databases (REMINDER)

Example script

Remove table (if exists)

```
DROP TABLE IF EXISTS TEST;
```

Create table with columns ID
and NAME

```
CREATE TABLE TEST(ID INT PRIMARY KEY,  
NAME VARCHAR(255));
```

Add new row

```
INSERT INTO TEST VALUES(1, 'Hello');
```

Add new row

```
INSERT INTO TEST VALUES(2, 'World');
```

Print table

```
SELECT * FROM TEST ORDER BY ID;
```

Change content of the row

```
UPDATE TEST SET NAME='Hi' WHERE ID=1;
```

Delete row

```
DELETE FROM TEST WHERE ID=2;
```

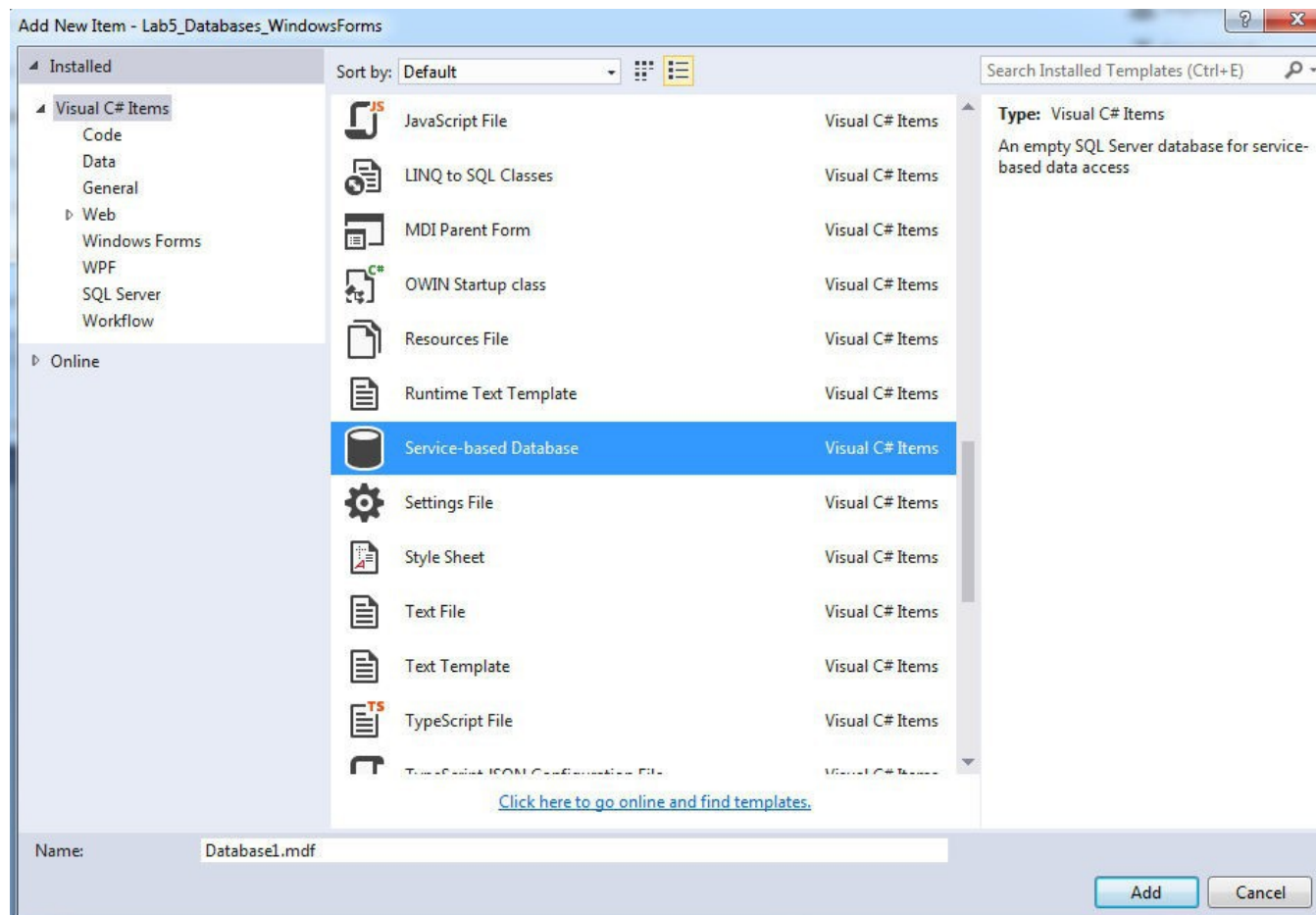




Create Database
Create Table
Add data

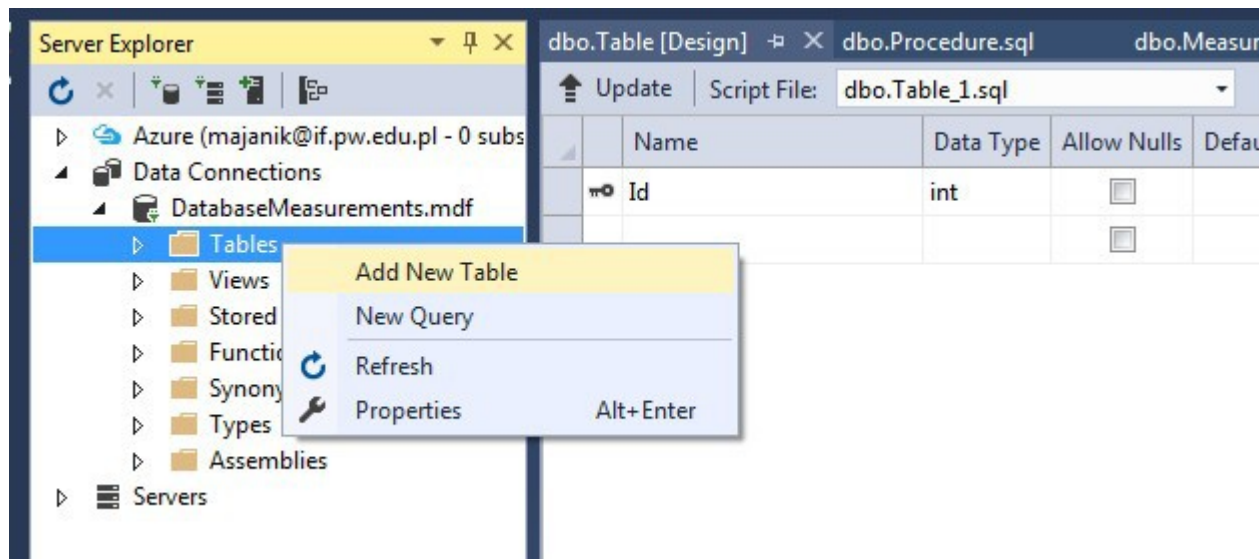
Create database

- New Console Application project
Click on the Project → New Item →
Service- based database (*.mdf file)



Create table

- Server Explorer (click two times on the database .mdf file in Solution Explorer) → Tables → Add New Table...

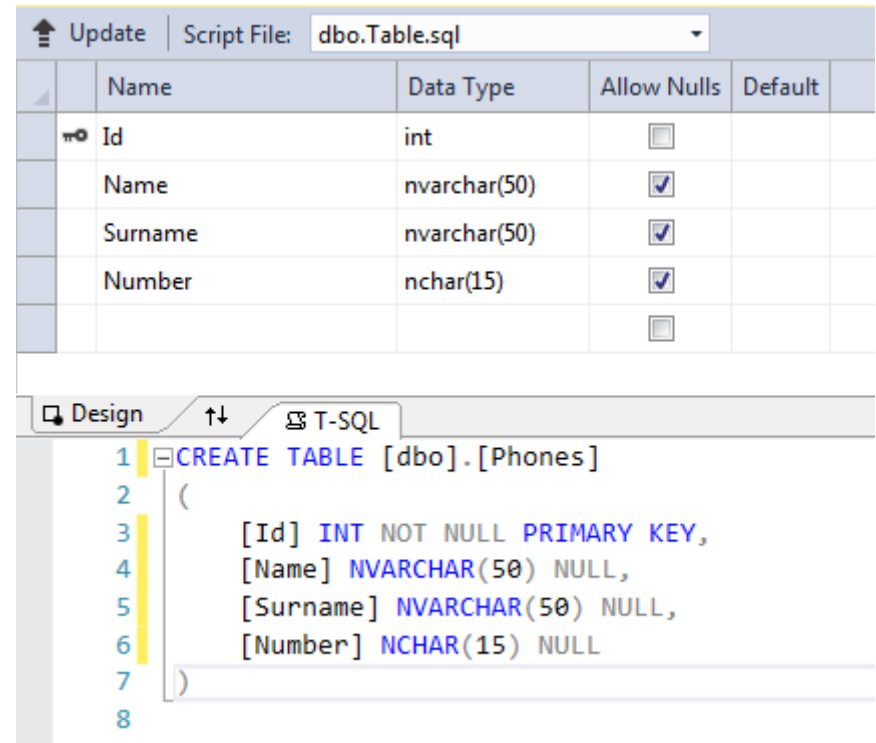
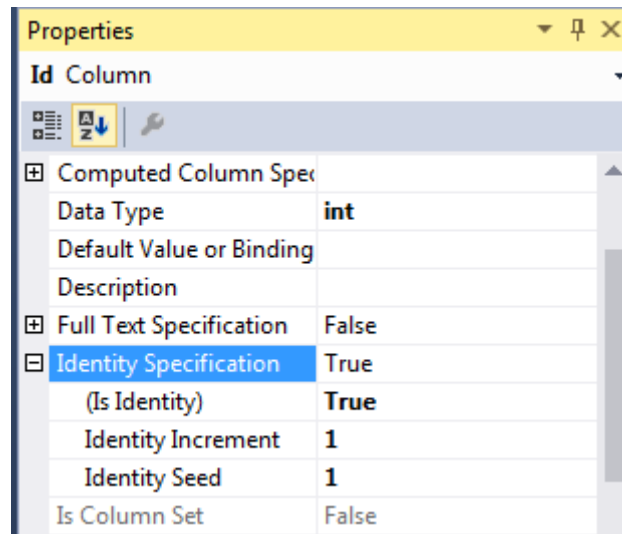


Create table

Create Table Phones with 4 columns:

- id (int)
- Name (nvarchar(50))
- Surname (nvarchar(50))
- Number (nchar(15))

Id is the Primary Key, should also be **identity** (check Column Properties)

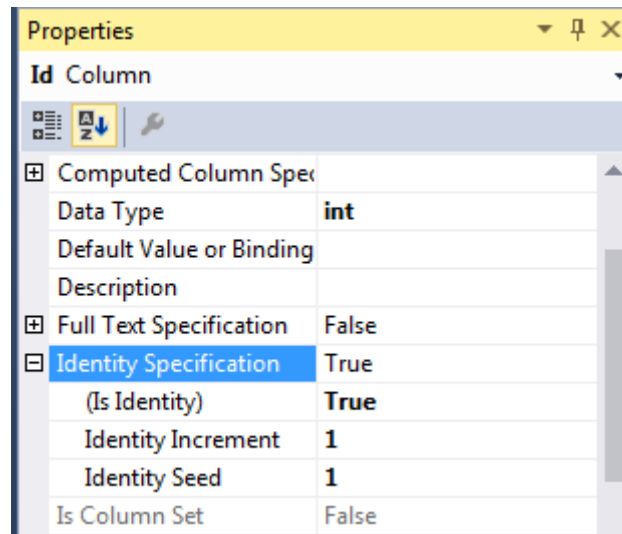


Create table

Create Table Phones with 4 columns:

- id (int)
- Name (nvarchar(50))
- Surname (nvarchar(50))
- Number (nchar(15))

Id is the Primary Key, should also be **identity** (check Column Properties)



The screenshot shows the 'Update' button in the SQL Server Enterprise Designer interface. A purple arrow points from the 'Update' button to the 'Identity Specification' section in the 'Properties' window.

Name	Data Type	Allow Nulls	Default
Id	int	☐	
time	datetime	☐	
value	real	☐	
comment	nvarchar(150)	☒	
		☐	

The screenshot shows the 'T-SQL' window in SQL Server Enterprise Designer. The following SQL statement is entered:

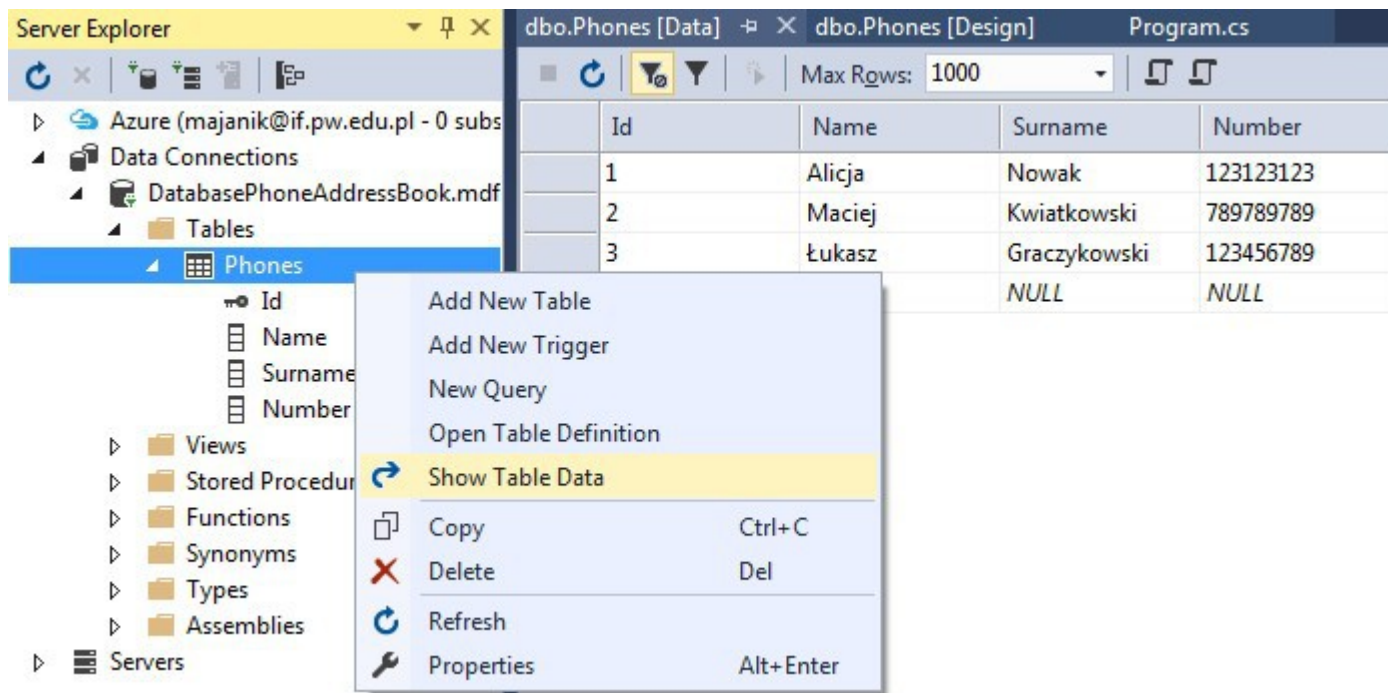
```
1 CREATE TABLE [dbo].[Measurements]
2 (
3     [Id] INT NOT NULL PRIMARY KEY IDENTITY,
4     [time] DATETIME NOT NULL,
5     [value] REAL NOT NULL,
6     [comment] NVARCHAR(150) NULL
7 )
8
```

Click „Update” when finished!
→ Update database

+ refresh the Tables folder in the
Server Explorer

Add data to database

- Click on this newly create table
 - Show Table Data
 - Add three example entries



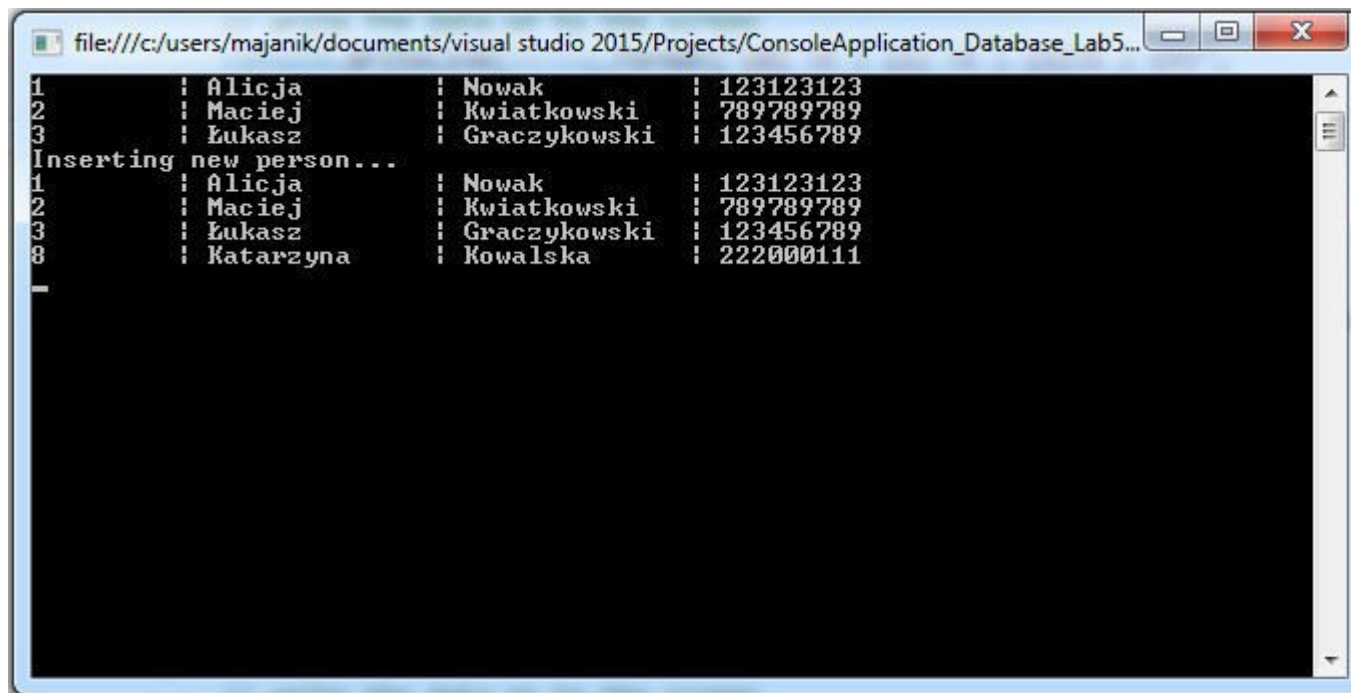


Show table
(Console Application)

Task: Console Application

- Print content of the Phones Table into the Console
- Insert one new entry
- Print Phones table again

3 pts!



```
file:///c:/users/majanik/documents/visual studio 2015/Projects/ConsoleApplication_Database_Lab5...
1      : Alicja      : Nowak      : 123123123
2      : Maciej       : Kwiatkowski : 789789789
3      : Łukasz      : Graczykowski : 123456789
Inserting new person...
1      : Alicja      : Nowak      : 123123123
2      : Maciej       : Kwiatkowski : 789789789
3      : Łukasz      : Graczykowski : 123456789
8      : Katarzyna   : Kowalska   : 222000111
```

Steps to accomplish Task 1

1. Create connection string

- Needed to connect your application to the database file

2. Establish connection

- Create connection (using connection string)
- Open connection

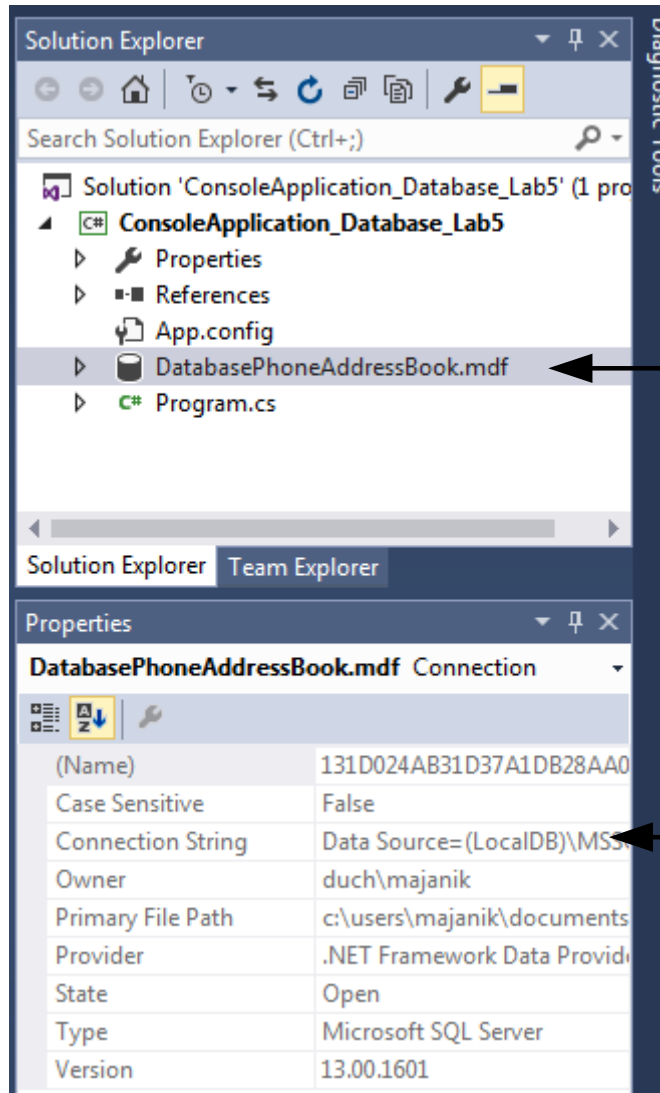
3. Prepare and execute commands (SQL statements)

4. Retrieve the data and print it on the screen

5. Remember to close connection! Or use “using” block

Connection string

Create connection string in the Main function. Copy the „Connection String” property from the database (delete all " characters as well as duplicate \ , see example below).



```
static void Main(string[] args)
{
    string connString = "Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=c:\users\majanik\documents\visual studio
2015\Projects\ConsoleApplication_Database_Lab5\ConsoleApplication_Database_Lab5\DatabasePhoneAddressBook.mdf;Integrated
Security=True";
}
```

Click two times

Copy this text („Connection String” property)

Basic SQL Statements

SQL commands needed for this task:

- **SELECT** - used when you want to read (or select) your data.
 - `SELECT * from TableName`
 - `SELECT Column1, Column2 from TableName`
- **INSERT** - used when you want to add (or insert) new data.
 - `INSERT INTO TableName VALUES (Value1, Value2 );`
`INSERT INTO TableName(Column2,Column3) VALUES (Value1,Value2);`

Connections, commands

- **Create connection:**

```
SqlConnection conn = new SqlConnection();  
conn.ConnectionString = connString;
```

- **Open connection:**

```
conn.Open();
```

- **Create SQL command:**

```
SqlCommand command = new SqlCommand("SELECT * FROM Phones", conn);  
                                Command                Connection
```

```
SqlCommand command2 = new SqlCommand("INSERT INTO Phones(Name,Surname,Number)  
VALUES ('Katarzyna','Kowalska', 222000111)", conn);
```

- **Execute command:**

```
command.ExecuteNonQuery();
```

Reading from SQL statements

- To read the data from the SQL statement we use the **SqlDataReader** available for the SqlCommand which returns the Reader object for the data. You can use this to read through the data and for each of the column provide the results on the screen.

```
SqlDataReader reader = command.ExecuteReader()
```

- Create new SqlDataReader object and read data from the command:

```
using (SqlDataReader reader = command.ExecuteReader())
{
    // while there is another record present
    while (reader.Read())
    {
        // write the data on to the screen
        Console.WriteLine(String.Format("{0} \t | {1} \t | {2} \t | {3}",
            // call the objects from their index
            reader[0], reader[1], reader[2], reader[3]));
    }
}
```

Using block

In C# there are some objects which use the resources of the system. Which need to be removed, closed, flushed and disposed etc. In C# you can either write the code to Create a new instance to the resource, use it, close it, flush it, dispose it. **Or you can simply just use using statement block in which the object created is closed, flushed and disposed automatically** and the resources are then allowed to be used again by other processes. This ensures that the framework would take the best measures for each process.

- Close & dispose

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString = "connection_string";
conn.Open();
// use the connection here
conn.Close();
```

- Use „using” block

```
using(SqlConnection conn = new SqlConnection())
{
    conn.ConnectionString = "connection_string";
    conn.Open();
    // use the connection here
}
```

<http://www.codeproject.com/Articles/823854/How-to-connect-SQL-Database-to-your-Csharp-program>



SQL Server + Windows Forms

Create new Windows Forms project...

Task: database with Windows Forms

- Create application to manage measurements.

Program should allow to:

- Insert new measurement (value, comment)
- Show measurements
- Update row
- Delete row

3 pts!

ID	Timestamp	Value
12	2016-11-11 16:55:28	453
13	2016-11-11 16:55:34	451
14	2016-11-11 16:56:36	452
15	2016-11-11 17:00:20	449
17	2016-11-12 13:05:54	448
18	2016-11-12 13:08:54	451
19	2016-11-12 13:09:28	451
20	2016-11-12 13:14:03	450
21	2016-11-12 13:24:25	532
22	2016-11-12 13:24:35	450
23	2016-11-12 13:24:48	451

Wrongly set parameters

Steps to accomplish Task 2

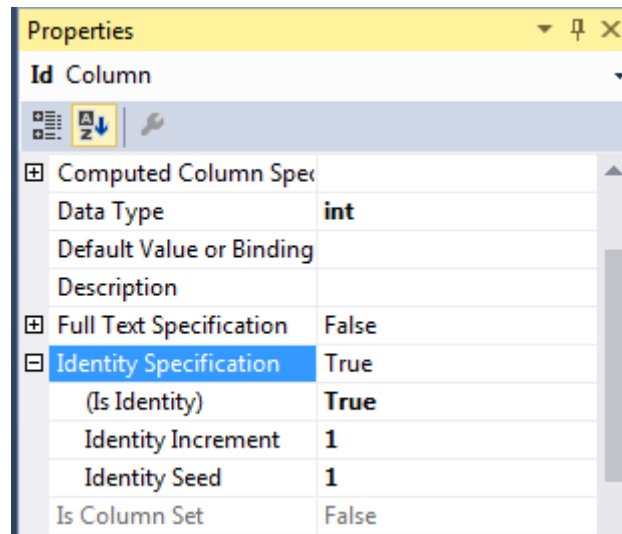
1. Create new database and table
2. Add all controls to the form
3. Program “insert” button
4. Program “update” and “delete” buttons

Prepare new database and table

Create Table Measurements with 4 columns:

- id (int)
- time (datetime)
- value (real)
- comment (nvarchar(150))

Only „comment” can be NULL.
Id is the Primary Key, should also be **identity** (check Column Properties)



Update Script File: dbo.Table.sql					
	Name	Data Type	Allow Nulls	Default	
	Id	int	☐		
	time	datetime	☐		
	value	real	☐		
	comment	nvarchar(150)	☒		
			☐		

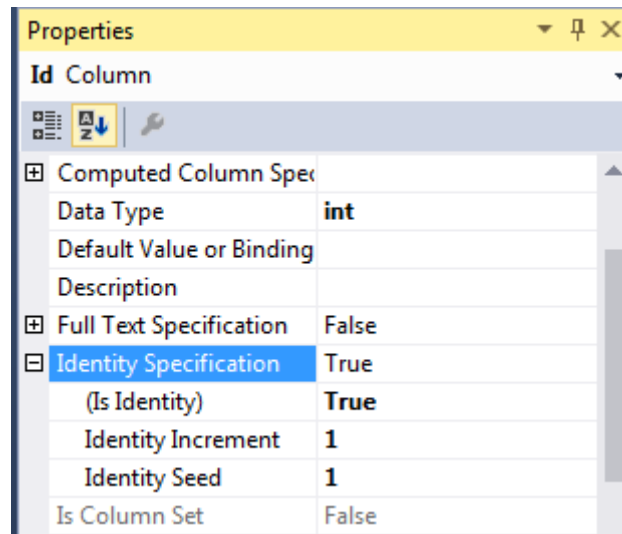
```
1 CREATE TABLE [dbo].[Measurements]
2 (
3     [Id] INT NOT NULL PRIMARY KEY IDENTITY,
4     [time] DATETIME NOT NULL,
5     [value] REAL NOT NULL,
6     [comment] NVARCHAR(150) NULL
7 )
8
```

Prepare new database and table

Create Table Measurements with 4 columns:

- id (int)
- time (datetime)
- value (real)
- comment (nvarchar(150))

Only „comment” can be NULL.
Id is the Primary Key, should also be **identity** (check Column Properties)



	Name	Data Type	Allow Nulls	Default
	Id	int	☐	
	time	datetime	☐	
	value	real	☐	
	comment	nvarchar(150)	☒	
			☐	

```
1 CREATE TABLE [dbo].[Measurements]
2 (
3     [Id] INT NOT NULL PRIMARY KEY IDENTITY,
4     [time] DATETIME NOT NULL,
5     [value] REAL NOT NULL,
6     [comment] NVARCHAR(150) NULL
7 )
8
```

REMEMBER TO
Click „Update” when finished!
→ Update database



Insert, Delete, Update

INSERT button

- Add 2 TextBox'es
- Add 3 Buttons
- Add 4 listBoxes

Value:

Insert

Delete

Update

Comment:

listBoxID	listBoxTime	listBoxValue	listBoxComment

- We start with INSERT button. When this button is clicked, we want to add to the database the content of the TextBox fields:
 - value – obligatory,
 - comment – not obligatorytogether with current datetime.

INSERT button

- Good practice: use try-catch close

```
try{  
    //All commands  
}  
catch (Exception ex)  
{  
    MessageBox.Show(ex.Message, "Error");  
}
```

- Show results in the ListBoxes
 - create „`private void loadlist()`” function that performs this task
 - Execute SELECT command and use SqlDataReader
 - Add new items to the ListBoxes
`listBoxID.Items.Add(datareader[0].ToString());`

INSERT button

- Advanced SQL commands:

```
SqlCommand sqlCmd = new SqlCommand();  
//set command text and parameters  
sqlCmd.CommandText = "INSERT INTO TableName (value, comment, time)  
VALUES (" + a + ", (@druga), (@time) )";
```

```
DateTime dateTimeVariable = DateTime.Now;  
sqlCmd.Parameters.AddWithValue("@time", dateTimeVariable);  
sqlCmd.Parameters.AddWithValue("@druga", "jakis tekst");  
sqlCmd.Connection = sqlCon; //set connection  
sqlCmd.ExecuteNonQuery(); //execute query
```

- In case of success:

```
MessageBox.Show("Saved successfully");
```

INSERT button

Result:

Measurement: Insert Delete Update

Value:

Comment:

12	2016-11-11 16:55:28	453	Wrongly set parameters
13	2016-11-11 16:55:34	451	
14	2016-11-11 16:56:36	452	
15	2016-11-11 17:00:20	449	
17	2016-11-12 13:05:54	448	
18	2016-11-12 13:08:54	451	
19	2016-11-12 13:09:28	451	
20	2016-11-12 13:14:03	450	
21	2016-11-12 13:24:25	532	
22	2016-11-12 13:24:35	450	
23	2016-11-12 13:24:48	451	

DELETE button

- Delete button should delete measurement with **Id selected in the ListBoxId**
- The DELETE statement is used to delete records in a table.
 - DELETE FROM table_name WHERE some_column=some_value;
- Suggestions
 - Again „sqlCmd.Parameters.AddWithValue” will be useful to set „some_value”
 - Use **listBoxID.SelectedItem** to extract Id value

UPDATE button

- Update button should update measurement with **Id selected in the ListBoxId using values from the TextBoxes**
- The UPDATE statement is used to update existing records in a table.
 - UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;



THE END

dr inż. Małgorzata Janik
malgorzata.janik@pw.edu.pl



Stored Procedures

```
CREATE PROCEDURE [dbo].[MeasurementAddOrEdit]
@mode nvarchar(10),
@Id int,
@time datetime,
@value real,
@comment nvarchar(150)

AS
if @mode='Add'
BEGIN
INSERT INTO Measurements (value, comment, time) VALUES
(@value, @comment, @time)
END
RETURN 0
```