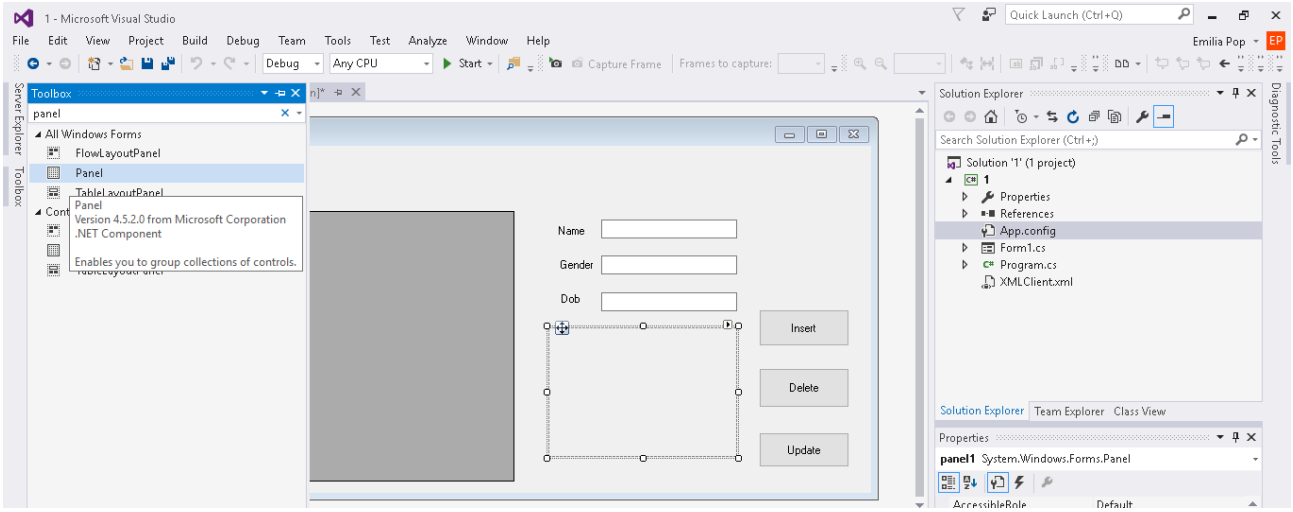
Lab 2

In this laboratory you have to generalize the Laboratory 1such that it will work for at least 2 scenarios (2 relations 1- n from your database).

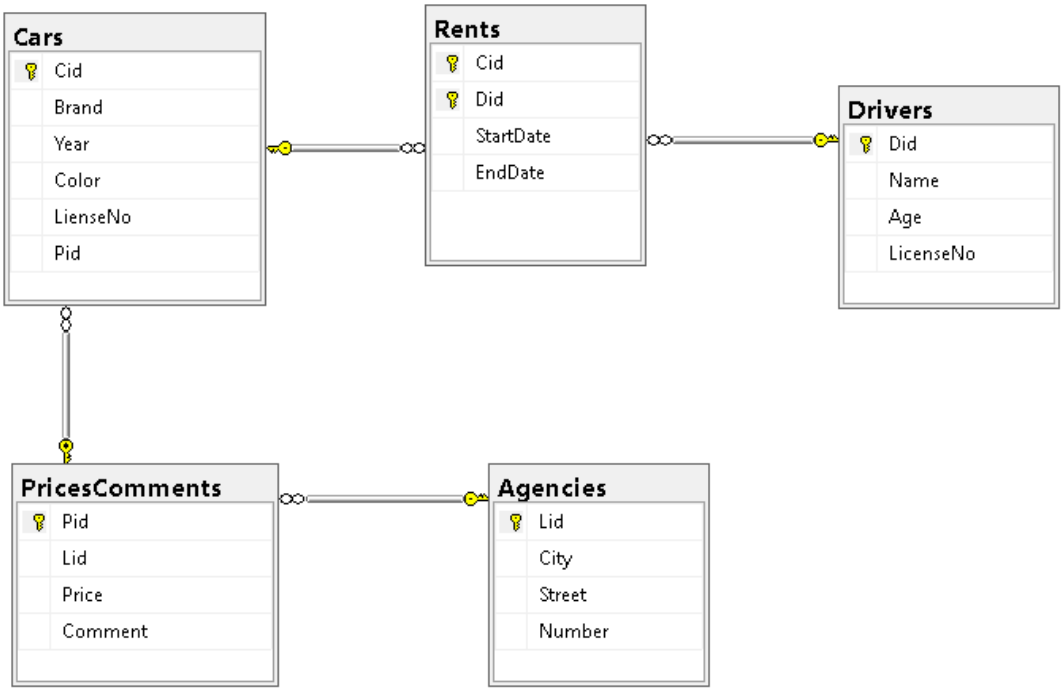
First, you have to consider a Panel in your form to create TextBoxes for the fields of the tables involved (for child tables in which the operations INSERT and UPDATE will use these TextBoxes for execution).



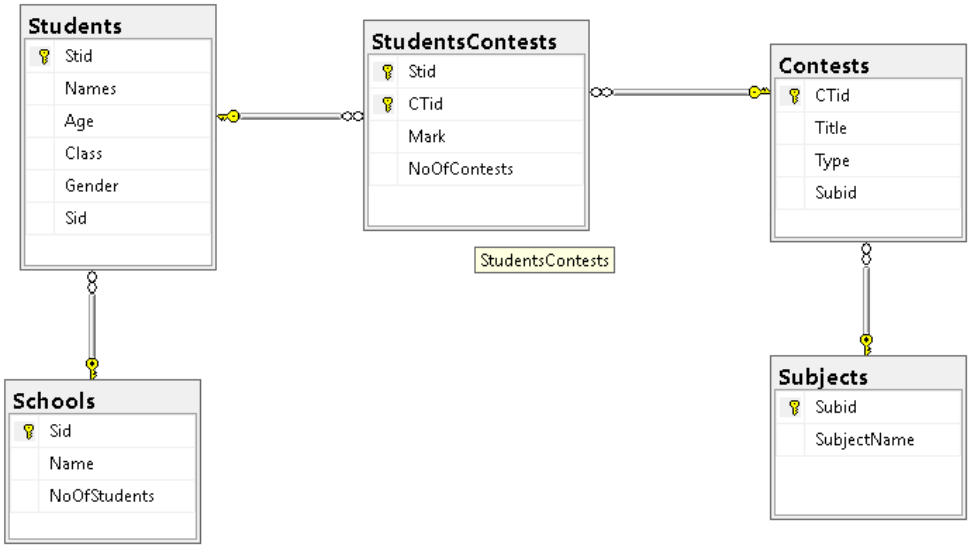
The TextBoxes from the Panel will be generated with code.
You need to put the Panel because not all the child tables have the same number of columns.

Example:

1. Agencies – PriceComments and PriceComments – Cars. The tables PriceComments and Cars are the child tables and do not have the same numbers of columns (for INSERT or UPDATE operations).

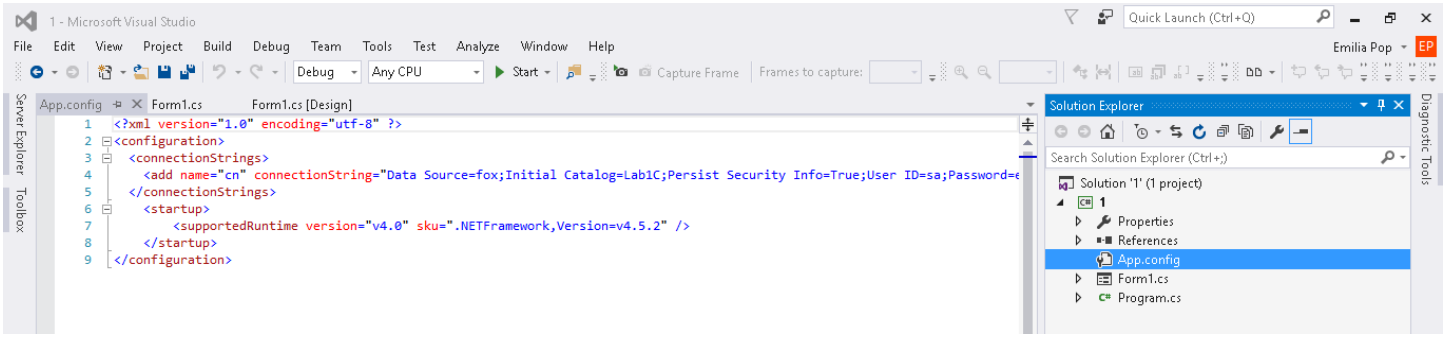


2. Schools – Students and Subjects – Contests. The tables Students and Constests are the child tables and have 6 fields, respectively 4 fields.

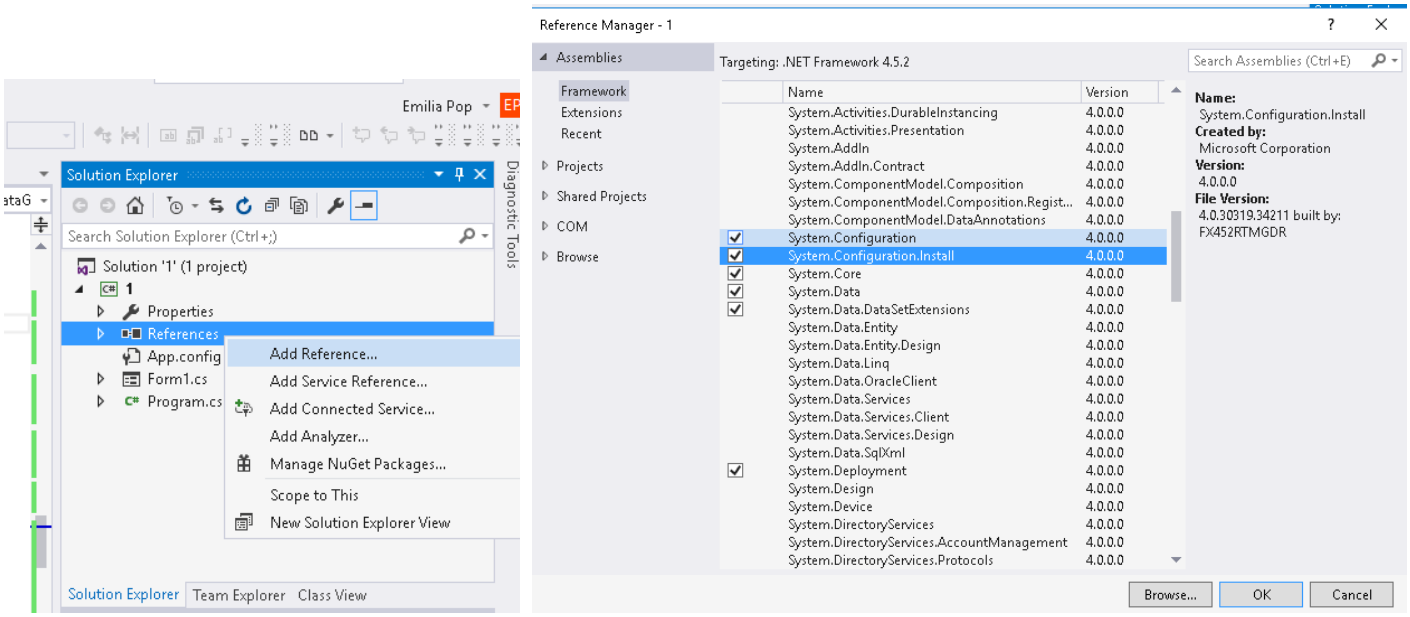


Everything that is related to tables and the CRUD operations (table name, fields, parameters, ...) will be set in an XML file.

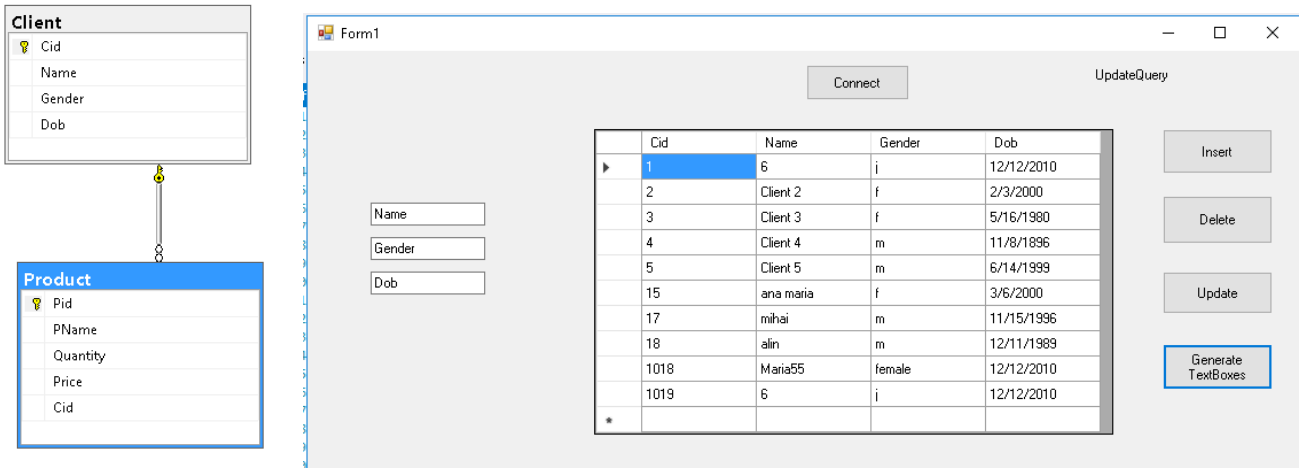
For example, you can work in App.config and use the „key" with the value. This one (the key) will be used in the source code. No change will be alowed in the source code when you will test the second relation 1-n. All the changes will be done in App.config.



In Form1.cs you have to include the package "using System.Configuration" for use the settings from App.config. If is not working, add this reference in the project.



Example: We consider the Client and Product tables.



Example of declare the connection and the select operation in App.config and use it in Form1.cs.

App.config	***Form1.cs***

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 	
<connectionStrings>
<add name= "cn" connectionString="Data Source=DESKTOP-ATJN5FL\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True"/>
</connectionStrings>
 	
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
</startup>
	SqlDataAdapter da = new SqlDataAdapter();
	DataSet ds = new DataSet();
	private void button1_Click(object sender, EventArgs e)
	{
	string con = ConfigurationManager.ConnectionStrings["cn"].ConnectionString;
	SqlConnection cs = new SqlConnection(con);
	string select = ConfigurationSettings.AppSettings["select"];
	da.SelectCommand = new SqlCommand(select, cs);
	//da.SelectCommand = new SqlCommand("Select * from Client", cs);
	ds.Clear();
	da.Fill(ds);
	dataGridView.DataSource = ds.Tables[0];

<appSettings> <add key= "select" value="Select * from Client" /> </appSettings> </configuration>	}

In App.config, one can define all the commands (select, insert, update, delete) or split the commands (name of table – parent / child; name of the columns; number of the columns; parameters..)

The "key" is the name used in the source code and the "value" is the command.

You have to prepare 2 scenarios for the tables involved in each of the 2 1-n relations. For example one of the scenarios can look like:

App.config	***Form1.cs***

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
…….
<appSettings>
<add key="ParentTableName" value="Client"/>
<add key="ChildTableName" value="Product"/>
<add key="ChildNumberOfColumns" value="3"/>
<add key="ChildColumnNames" value="PName,Quantity,Price"/>
<add key="ColumnNamesInsertParameters" value ="@pname,@quantity, @price"/>
<add key ="UpdateQuery" value="UPDATE Product SET PName = @pname, Quantity=@quantity, Price=@price WHERE Cid=@cid"/>
……
</appSettings>
……..
</configuration>
	…
	string ChildTableName = ConfigurationManager.AppSettings["ChildTableName"];
	string ChildColumnNames = ConfigurationManager.AppSettings["ChildColumnNames"];
	string ColumnNamesInsertParameters = ConfigurationManager.AppSettings["ColumnNamesInsertParameters"];
	List<string> ColumnNamesList = new List<string>(ConfigurationManager.AppSettings["ColumnNames"].Split(','));
	SqlCommand cmd = new SqlCommand("INSERT INTO " + ChildTableName + " (" + ChildColumnNames + ") VALUES (" + ColumnNamesInsertParameters + ")", cs);
	foreach (string column in ColumnNamesList) {
	TextBox textBox = (TextBox)panel1.Controls[column];
	cmd.Parameters.AddWithValue("@" + column, textBox.Text); }
	cs.Open();
	cmd.ExecuteNonQuery();
	ds.Clear();
	da.Fill(ds);
	cs.Close();

Here the Insert operation was done like a concatenation.

It is up to you if you define the operations directly in App.config or just the elements that are involved (like, name of the tables, columns, ….). The ***value*** will change.

In Form1.cs nothing will be changed.

A second scenario can look like

App.config	***App.config***
<?xml version="1.0" encoding="utf-8" ?> <configuration> ……. <appSettings> <add key="ParentTableName" value="Client"/> <add key="ChildTableName" value="Product"/> <add key="ChildNumberOfColumns" value="3"/> <add key="ChildColumnNames" value="PName,Quantity,Price"/> <add key="ColumnNamesInsertParameters" value ="@pname,@quantity, @price"/> <add key ="UpdateQuery" value="UPDATE Product SET PName = @pname, Quantity=@quantity, Price=@price WHERE Cid=@cid"/> …… </appSettings> …….. </configuration>	<?xml version="1.0" encoding="utf-8" ?> <configuration> ……. <appSettings> <add key="ParentTableName" value="Categorii"/> <add key="ChildTableName" value="Produse"/> <add key="ChildNumberOfColumns" value="1"/> <add key="ChildColumnNames" value="nume"/> <add key="ColumnNamesInsertParameters" value ="@nume"/> <add key ="UpdateQuery" value="UPDATE Produse SET nume=@nume"/> …… </appSettings> …….. </configuration>