Laborator – EF – Model Design First

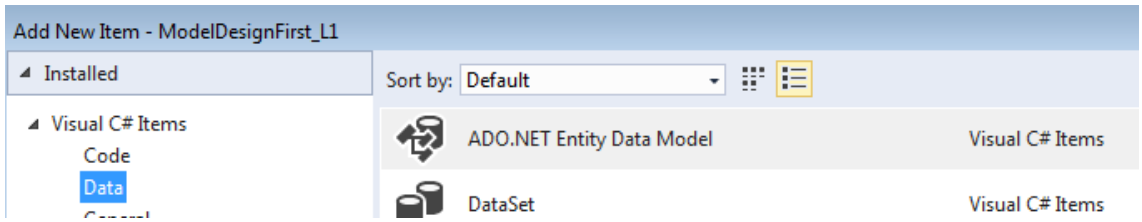
Verificati ca aveti serverul SQL pornit.

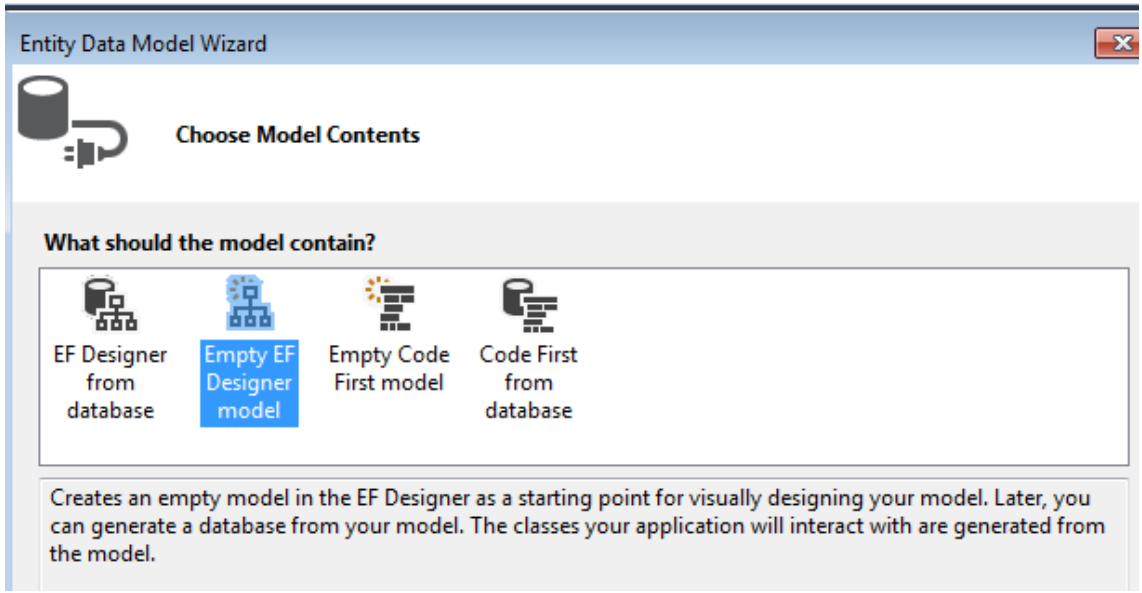
In servicii ar trebui sa aveti ceva de genul:

SQL Server (SQL2012NEW)	Provides sto...	Started	Automatic	NT Service...
SQL Server (SQLEXPRESS)	Provides sto...	Started	Automatic	NT Service...
SQL Server Agent (SQL2012)	Executes jo...		Disabled	Local Syste...
SQL Server Agent (SQL2012NEW)	Executes jo...	Started	Manual	NT Service...
SQL Server Agent (SQLEXPRESS)	Executes jo...		Automatic	Network S...
SQL Server Analysis Services (S...	Supplies onl...		Disabled	Local Syste...
SQL Server Analysis Services (S...	Supplies onl...	Started	Automatic	NT Service...
SQL Server Browser	Provides SQ...	Started	Automatic (D...	Local Service

Figure 1

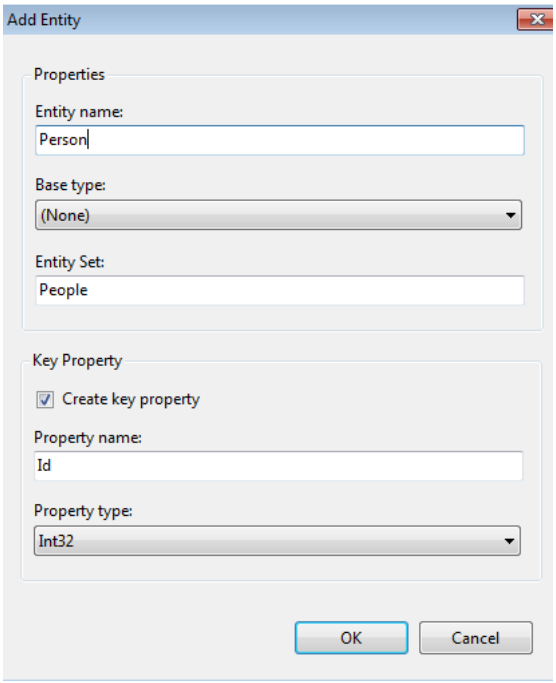
Etape de urmat in mediul de dezvoltare VS 2015:

1. Cream un proiect “Console Application” cu numele ModelDesignFirst_L1.
2. Adaugam un articol nou
3. 

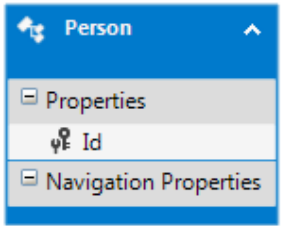
4. Numele este **Model1**
5. Selectam **Empty EF Designer model**
6. 

7. **Finish**

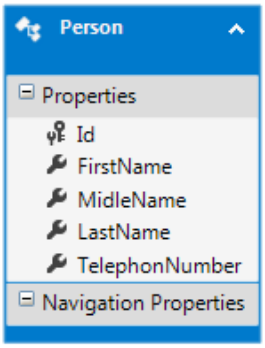
8. Pe suprafata de lucru ce apare facem clic dreapta si selectam ***Add New -> Entity*** ca in figura



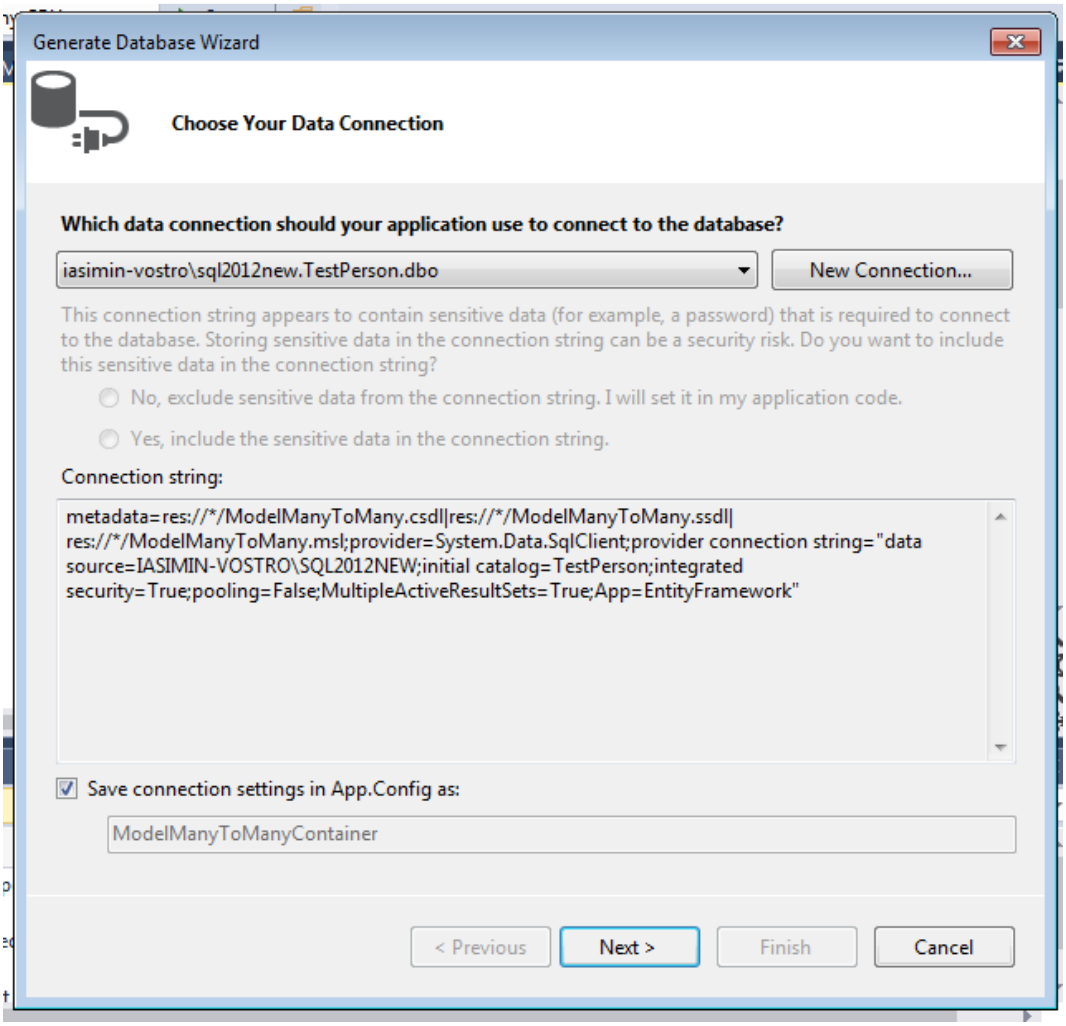
9.
10. Numele tabelei din baza de date va fi *People*. Folosind meniul contextual pe figura



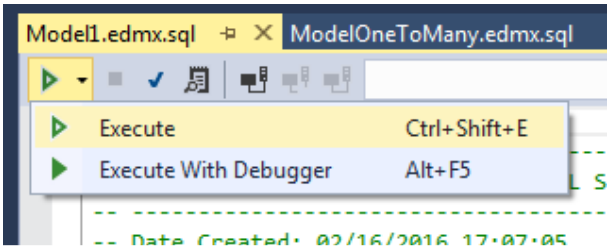
11. Adaugam proprietatile scalare: *FirstName* , *MidleName*, *LastName* si *TelephonNumber* toate de tip string cu lungimea maxima de **10** caractere (vedeti **Properties** pe fiecare proprietate adaugata). In final avem:



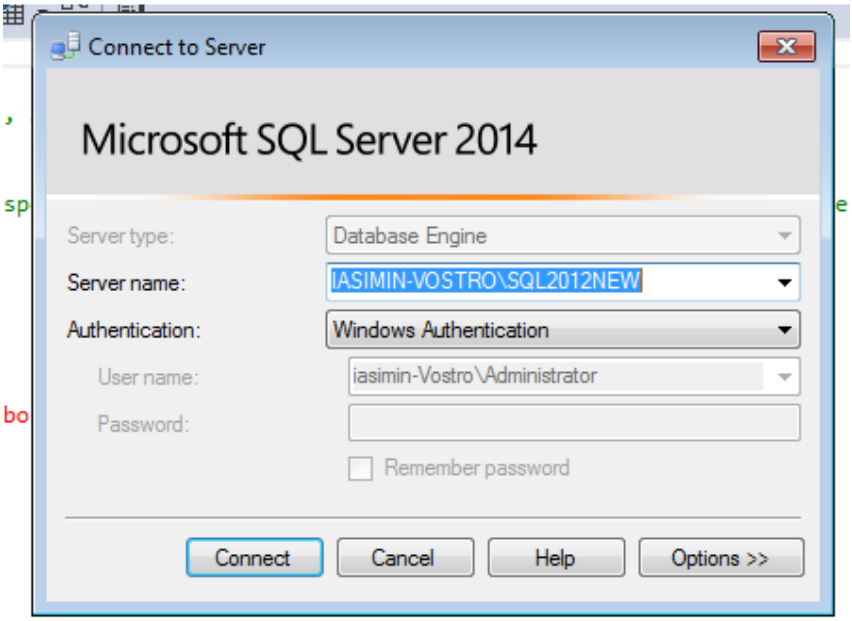
12. **ATENTIUNE!** Pe Server Explorer din VS 2015 cream o baza de date cu numele ***TestPerson***. Serverul SQL trebuie sa fie pornit inainte de a executa aceasta actiune. Tablea generata va fi in aceasta baza de date. In acest moment baza de date nu contine tabele definite.
13. De pe suprafata de lucru, meniu contextual selectam *Generate Database from Model*. In acest moment va trebui sa furnizam baza de date creata anterior. Salvam.



Optiunea Generate Database from Model creaza numai un script ce contine DDL (Database Definition Language) si nu creaza baza de date si nu va executa automat acest script. Noi ca dezvoltatori, va trebui sa executam acest script. Putem modifica acest script si apoi sa-l executam.



Va trebui apoi sa ne conectam la SQL Server



14. Verificam ce s-a generat in cadrul proiectului. Identificam contextul si clasa *Person*. Verificam ce avem in fisierul de configurare. Ceva asemanator ar trebui sa aveti:

```
<connectionStrings>
    <add name="Model1Container"
connectionString="metadata=res://*/Model1.csdl|res://*/Model1.ssdl|r
es://*/Model1.msl;provider=System.Data.SqlClient;provider connection
string=&quot;data source=IASIMIN-VOSTRO\SQL2012NEW;initial
catalog=TestPerson;integrated
security=True;pooling=False;MultipleActiveResultSets=True;App=Entity
Framework&quot;" providerName="System.Data.EntityClient" />
    <add name="ModelOneToManyContainer"
  </connectionStrings>
```

Tabela *People* are urmatoarea structura (generata de utilitarul din VS 2015):

```
CREATE TABLE [dbo].[People] (
    [Id]             INT           IDENTITY (1, 1) NOT NULL,
    [FirstName]      NVARCHAR (10) NOT NULL,
    [MidleName]      NVARCHAR (10) NOT NULL,
    [LastName]       NVARCHAR (10) NOT NULL,
    [TelephonNumber] NVARCHAR (10) NOT NULL,
    CONSTRAINT [PK_People] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Clasa *Person* are urmatoarea definitie:

```
public partial class Person
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
```

```
        public string MidleName { get; set; }
        public string LastName { get; set; }
        public string TelephonNumber { get; set; }
    }
```

Observatie
Proprietatile din *Person* nu au atribute care sa specifice marimea unui string. Aceste informatii se gasesc in fisierul ce descrie modelul conceptual, de memorare si de mapare.

In cazul nostru in fisierul ce descrie modelul **conceptual** (.csdl) gasim urmatoarele definitii (subdirectorul obj/Release sau obj/Debug):

```
...
<EntityContainer Name="Model1Container"
annotation:LazyLoadingEnabled="true">
    <EntitySet Name="People" EntityType="Model1.Person" />
  </EntityContainer>
  <EntityType Name="Person">
    <Key>
      <PropertyRef Name="Id" />
    </Key>
    <Property Name="Id" Type="Int32" Nullable="false"
       annotation:StoreGeneratedPattern="Identity" />
    <Property Name="FirstName" Type="String" Nullable="false"
            MaxLength="10" />
    <Property Name="MidleName" Type="String" Nullable="false"
            MaxLength="10" />
    <Property Name="LastName" Type="String" Nullable="false"
            MaxLength="10" />
    <Property Name="TelephonNumber" Type="String" Nullable="false"
            MaxLength="10" />
  </EntityType>
```

Modelul de **mapare** (.msl) contine:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping Space="C-S"
xmlns="http://schemas.microsoft.com/ado/2009/11/mapping/cs">
  <EntityContainerMapping
StorageEntityContainer="Model1StoreContainer"
CdmEntityContainer="Model1Container">
    <EntitySetMapping Name="People">
      <EntityTypeMapping TypeName="IsTypeOf(Model1.Person)">
        <MappingFragment StoreEntitySet="People">
          <ScalarProperty Name="Id" ColumnName="Id" />
          <ScalarProperty Name="FirstName" ColumnName="FirstName" />
          <ScalarProperty Name="MidleName" ColumnName="MidleName" />
          <ScalarProperty Name="LastName" ColumnName="LastName" />
          <ScalarProperty Name="TelephonNumber"
                ColumnName="TelephonNumber" />
        </MappingFragment>
      </EntityTypeMapping>
    </EntitySetMapping>
  </EntityContainerMapping>
```

```
</Mapping>
```

Observati maparea dintre tabela People si clasa Person:

```
<EntitySetMapping Name="People">
      <EntityTypeMapping TypeName="IsTypeOf(Model1.Person)">
```

Sunt descrise si maparile dintre proprietati si coloanele din tabela.

Modelul de **memorare** (descris in fisierul .ssdl) are urmatorul continut:

```
<?xml version="1.0" encoding="utf-8"?>
<Schema Namespace="Model1.Store" Alias="Self" Provider="System.Data.SqlClient"
ProviderManifestToken="2012"
xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
xmlns="http://schemas.microsoft.com/ado/2009/11/edm/ssdl">
  <EntityContainer Name="Model1StoreContainer">
    <EntitySet Name="People" EntityType="Model1.Store.People" store:Type="Tables" Schema="dbo" />
  </EntityContainer>
  <EntityType Name="People">
    <Key>
      <PropertyRef Name="Id" />
    </Key>
    <Property Name="Id" Type="int" StoreGeneratedPattern="Identity" Nullable="false" />
    <Property Name="FirstName" Type="nvarchar" Nullable="false" MaxLength="10" />
    <Property Name="MidleName" Type="nvarchar" Nullable="false" MaxLength="10" />
    <Property Name="LastName" Type="nvarchar" Nullable="false" MaxLength="10" />
    <Property Name="TelephonNumber" Type="nvarchar" Nullable="false" MaxLength="10" />
  </EntityType>
</Schema>
```

1. In Fisierul ce contine Main scriem urmatorul cod

```
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Test Model Designer First");
            TestPerson();

            Console.ReadKey();
        }

        static void TestPerson()
        {
            using (Model1Container context = new Model1Container())
            {
                Person p = new Person() { FirstName = "Julie",
                    LastName = "Andrew", MidleName = "T",
                           TelephonNumber = "1234567890" };
                context.People.Add(p);
                context.SaveChanges();
                var items = context.People;
                foreach (var x in items)
                    Console.WriteLine("{0} {1}", x.Id, x.FirstName);
            }
        }
       }
```

Testam.

Modificam codul astfel incat sa introducem valori de la tastatura.

In cadrul aceluaisi proiect continuam cu:

Creare model ce contine asocieri de tipul 1->*

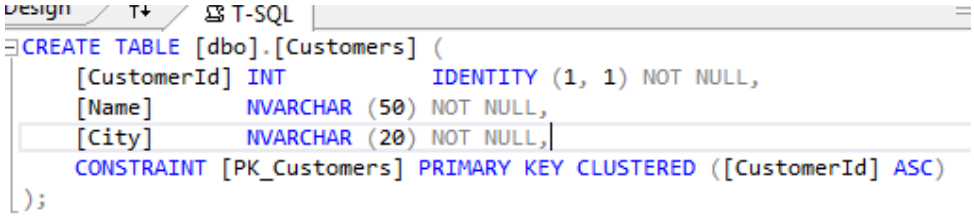
II. Cream in aceeasi baza de date tabele ce sunt in relatia 1->*. Tabela *Customer* si *Order*.

Tipul Customer are proprietatile:

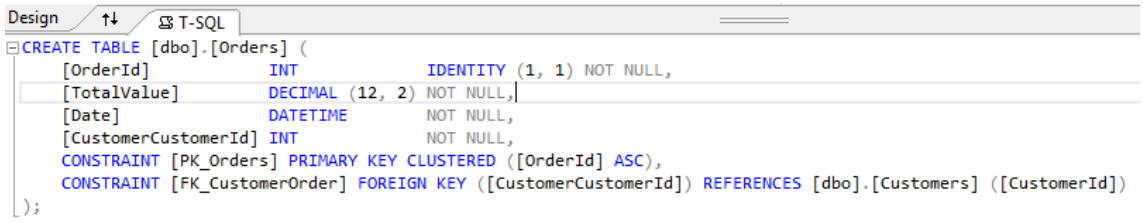
```
public int CustomerId { get; set; }
public string Name { get; set; }
public string City { get; set; }
```

unde *CustomerId* este primary key in tabela ce va fi generata.

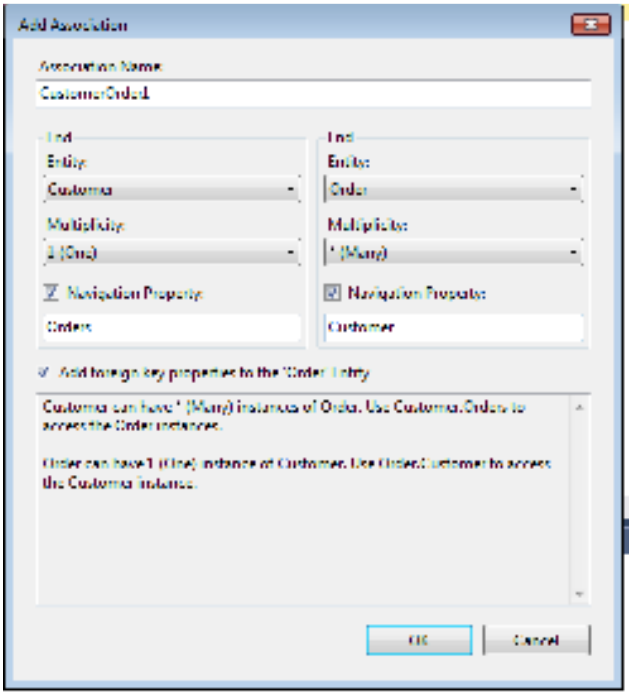
Structura tabelei *Customers* este:



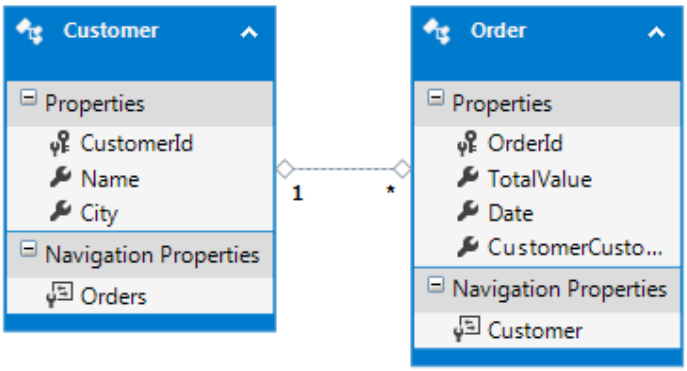
Structura tabelei *Orders* este:



Stabilirea relatiei de asociere (meniu contextual pe suprafata de lucru si adaugare asociere) :



Dupa ce le-am creat adaugam relatia de **asociere**. Vedeti meniul contextual. In final trebuie sa avem ceva de genul:



Acesta e contextul

```
public partial class ModelOneToManyContainer : DbContext
 {
     public ModelOneToManyContainer()
         : base("name=ModelOneToManyContainer")
     { }

     protected override void OnModelCreating(
                 DbModelBuilder modelBuilder)
     {
         throw new UnintentionalCodeFirstException();
      }
```

```
        public virtual DbSet<Customer> Customers { get; set; }
        public virtual DbSet<Order> Orders { get; set; }
       }
```

Generam tabelele si selectam **aceeasi** baza de date. In acest moment in cadrul aplicatiei avem doua modele si doua contexte diferite. Verificam ce s-a generat in cadrul proiectului pentru noul model.

Cream o metoda statica *TestOneToMany*() ce va fi apelata din Main in vederea testarii modelului.

Codul poate fi:

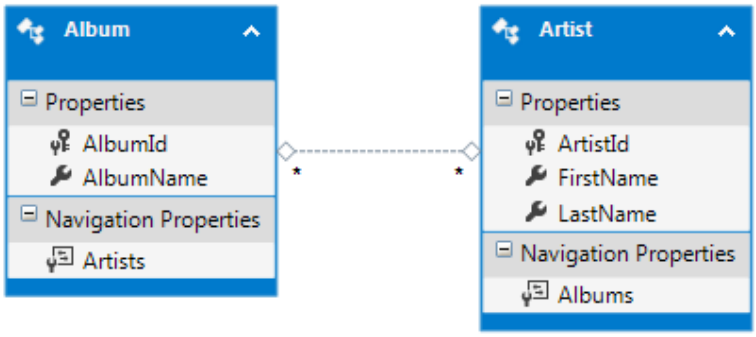
```
static void TesTOneToMany()
        {
            Console.WriteLine("One to many association");
            using (ModelOneToManyContainer context =
                    new ModelOneToManyContainer())
            {
                Customer c = new Customer() { Name = "Customer 1",
                    City = "Iasi" };
                Order o1 = new Order() { TotalValue = 200,
                    Date = DateTime.Now, Customer = c };
                Order o2 = new Order() { TotalValue = 300,
                    Date = DateTime.Now, Customer = c };
                context.Customers.Add(c);
                context.Orders.Add(o1);
                context.Orders.Add(o2);
                context.SaveChanges();

                var items = context.Customers;
                foreach(var x in items)
                {
                    Console.WriteLine("Customer : {0}, {1}, {2}",
                          x.CustomerId, x.Name, x.City);
                    foreach(var ox in x.Orders)
                        Console.WriteLine("\tOrders: {0}, {1}, {2}",
                          ox.OrderId, ox.Date, ox.TotalValue);
                }
            }
        }
```

Modificati codul astfel incat sa introducem date de la tastatura.

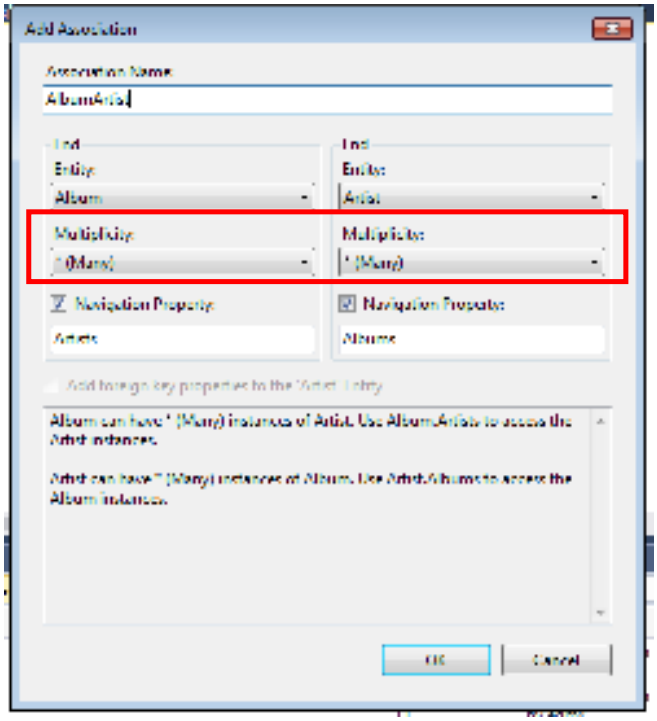
III. Relatie **many-to-many.** Tabela de legatura nu contine informatii suplimentare.

Model Designer First: Name este **ModelManyToMany**.

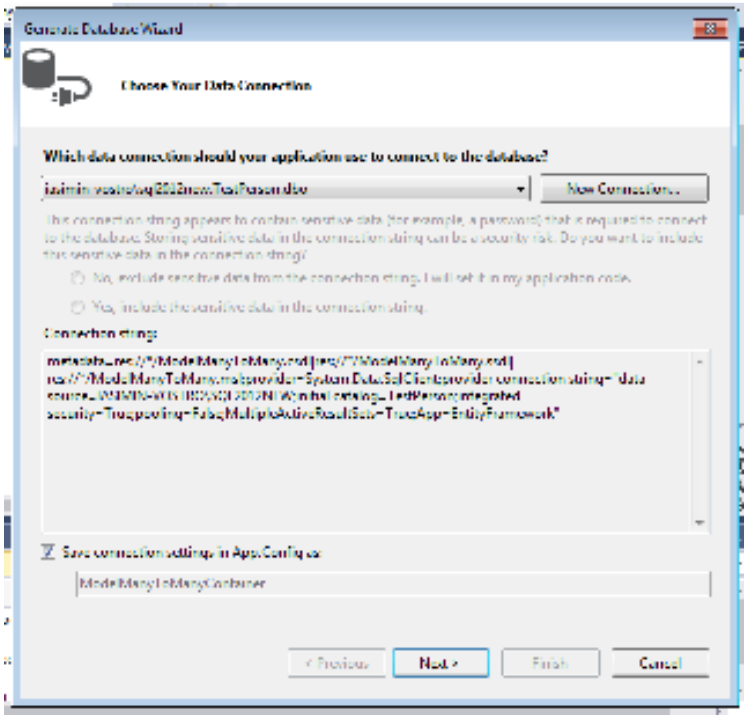
Se creaza entitatile ca in figura de mai jos:



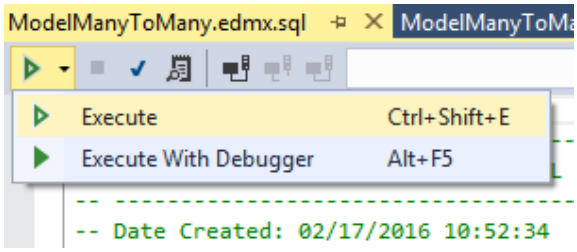
Cand se adauga asocierea trebuie sa avem ceva de genul:



Generam baza din model si selectam baza de date pe care am creat-o anterior si apoi Next si Finish.



Executam scriptul. Se va cere conectarea la SQL Server.



Verificam. In baza de date s-au creat tabelele **Artists**, **Albums** si **AlbumArtist** (tabela de legatura).

Cod pentru testare.

IV. Relatie many-to-may si tabela de legatura contine informatii suplimentare.

In acest caz Model Designer First nu poate fi folosit. Se foloseste modelul de programare *Database First*.