The Learning House

427 S. 4th Street #300

Louisville KY 40202

This image cannot currently be displayed.

Entity Framework EDMX Designer

.NET Cohort

Lesson Goals

- Learn how the designer translates the conceptual model to a class diagram
- Learn how to update the EDMX
- Learn how to manipulate the Mapping Details for fine-grained control over the map between the model and the database schema
- Learn how to use the Model Browser to explore the various Entity Framework Models.

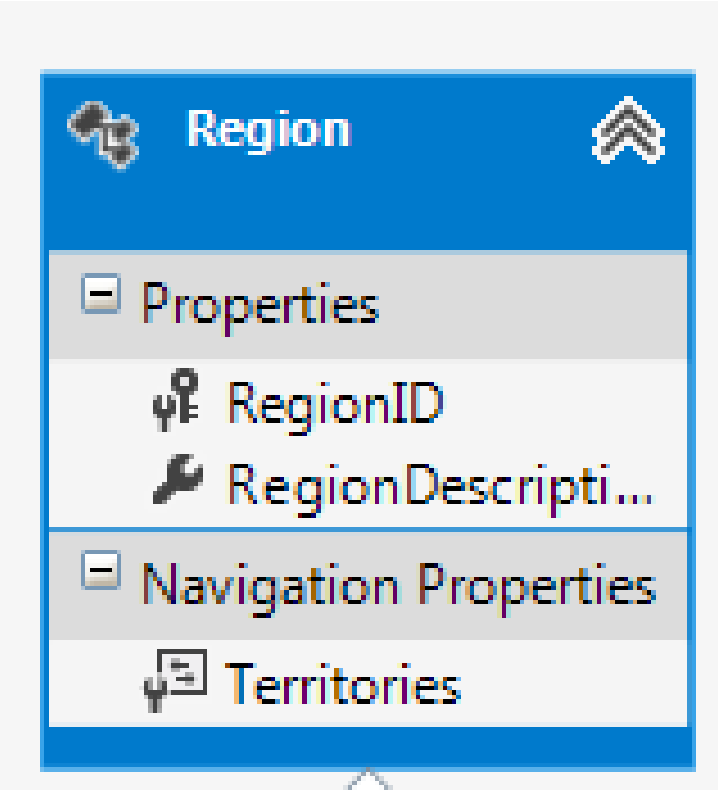
SOFTWARE GUILD

The Designer and the EDMX

The Entity Type definitions are represented as entity classes on the designer surface.

The EDMX file is XML-formatted behind the designer, and keeps track of all the mappings and relationships from the database to the class properties. (Navigation/Associations)

This can all be seen in the properties window for a selected entity.



Region Table's EDMX Mappings

```
<EntitySetMapping Name="Regions">
  <EntityTypeMapping TypeName="NorthwindModel.Region">
    <MappingFragment StoreEntitySet="Region">
      <ScalarProperty Name="RegionID" ColumnName="RegionID" />
      <ScalarProperty Name="RegionDescription" ColumnName="RegionDescription" />
    </MappingFragment>
  </EntityTypeMapping>
</EntitySetMapping>

<Association Name="FK_Territories_Region">
  <End Role="Region" Type="NorthwindModel.Region" Multiplicity="1" />
  <End Role="Territories" Type="NorthwindModel.Territory" Multiplicity="*" />
  <ReferentialConstraint>
    <Principal Role="Region">
      <PropertyRef Name="RegionID" />
    </Principal>
    <Dependent Role="Territories">
      <PropertyRef Name="RegionID" />
    </Dependent>
  </ReferentialConstraint>
```

SOFTWARE GUILD

Viewing Details on the Model

We can right-click an entity and select “Properties” to make changes to the class itself.

We can also select individual fields/columns as well as Navigation properties and associations (the lines between entities).

Code Generation	
Abstract	False
Access	Public
Diagram	
Fill Color	0, 122, 204
General	
Base Type	(None)
Documentation	
Long Description	
Summary	
Entity Set Name	**Regions**
Name	**Region**

Code Generation	
Getter	Public
Setter	Public
General	
Concurrency Mode	None
Default Value	(None)
Documentation	
Entity Key	**True**
Name	**RegionID**
Nullable	**False**
StoreGeneratedPatter	None
Type	**Int32**

Code Generation	
Getter	Public
Setter	Public
General	
Documentation	
Multiplicity	*** (Many)**
Name	**Territories**
Return Type	Collection of Territory
Navigation	
Association	**FK_Territories_Region**
From Role	Region
To Role	Territories

Constraints	
Referential Constraint	Region -> Territory
General	
Association Set Name	**FK_Territories_Region**
Documentation	
End1 Multiplicity	**1 (One of Region)**
End1 Navigation Prop	**Territories**
End1 OnDelete	None
End1 Role Name	Region
End2 Multiplicity	*** (Collection of Territo**
End2 Navigation Prop	**Region**
End2 OnDelete	None
End2 Role Name	**Territories**
Name	**FK_Territories_Region**

Making Changes to the Database

- Let's make some changes to Pubs in the server explorer.
  - In the customers table, let's change ContactTitle to Title and delete the Fax.
- Notice, when we go back to the model, the updates aren't there. We have to update the EDMX!

SOFTWARE GUILD

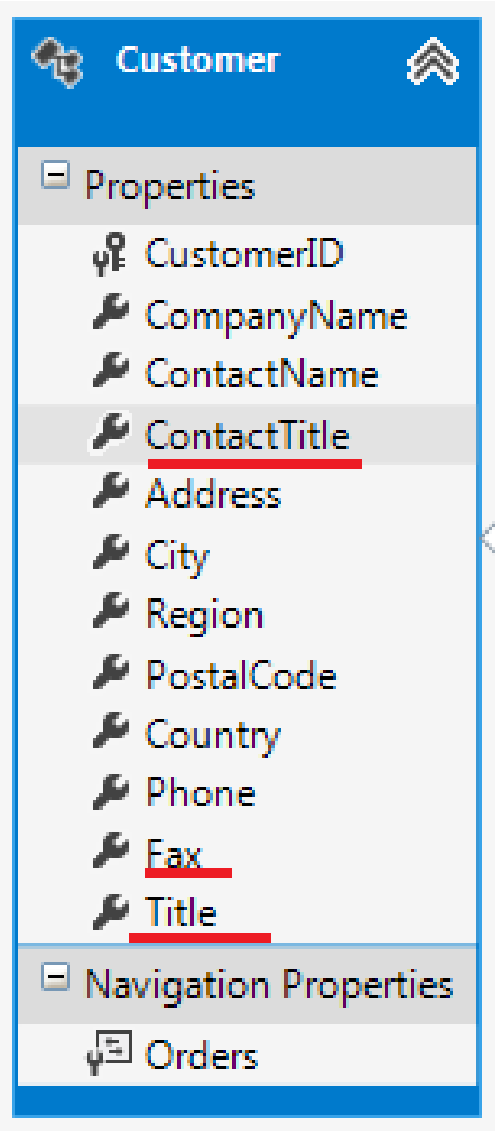
Updating the model

On the EDMX designer surface, right-click and select "Update Model from Database"

Notice, in the Refresh tab, there are a lot more tables selected than what we made changes to. Because of the relationships, Visual Studio thinks any table that is related (directly or indirectly) needs to be refreshed.

A couple errors pop up, we notice Title has been added, but the other two fields remain.

What gives?



What Gives?

- There is a relationship between the EDMX, the Database, and the Designer.
  - The CSDL (Conceptual Schema Definition Language) is the conceptual model that is represented by your code objects in the primary designer window.
  - The SSDL (Store Schema Definition Language) represents the database. You can see this in the Model Browser.
  - The MSL (Mapping Schema Language) controls the relationship between the CSDL and the SSDL.

That’s Nice, but Seriously, What Gives?

- All the wizard does is update the SSDL to reflect the database changes and update the MSL so that nothing in the conceptual model was mapped to nonexistent fields. Otherwise, it left the CSDL model as we designed it.
- So let’s right-click on the design surface and choose Mapping Details, then select the Customers entity.

Moral of the Story

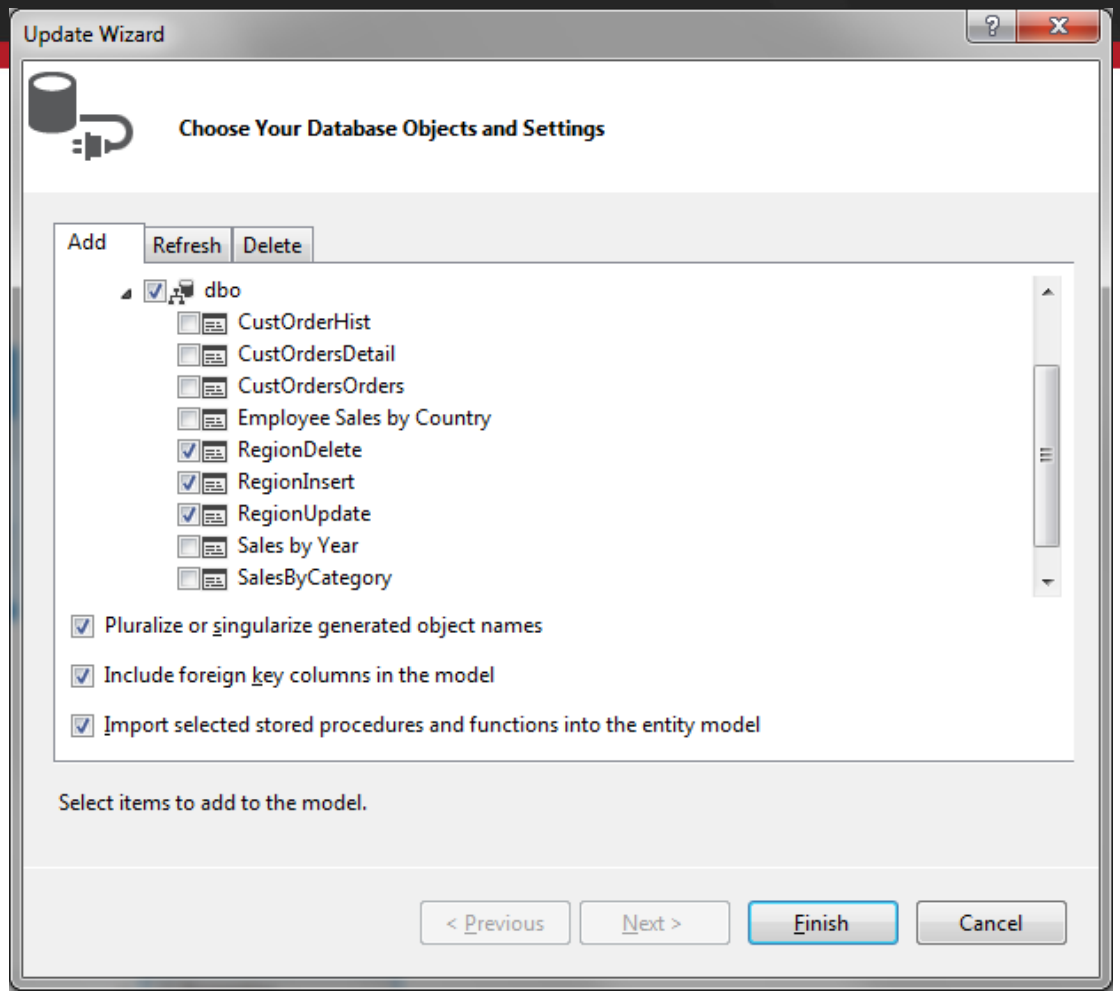
There is no way (or at least no practical way) for the wizard to know which of the changes you've made to the conceptual model you want to keep. It makes the changes it can—to the schema definition and the mapping layer—and leaves the decisions it can't make to the person who can (you). So, suck it up and delete the fields by hand.

Column Mappings		
CustomerID : nchar	↔	CustomerID : String
CompanyName : nvarchar	↔	CompanyName : String
ContactName : nvarchar	↔	ContactName : String
Title : nvarchar	↔	Title : String
Address : nvarchar	↔	Address : String
City : nvarchar	↔	City : String
Region : nvarchar	↔	Region : String
PostalCode : nvarchar	↔	PostalCode : String
Country : nvarchar	↔	Country : String
Phone : nvarchar	↔	Phone : String

Mapping Functions

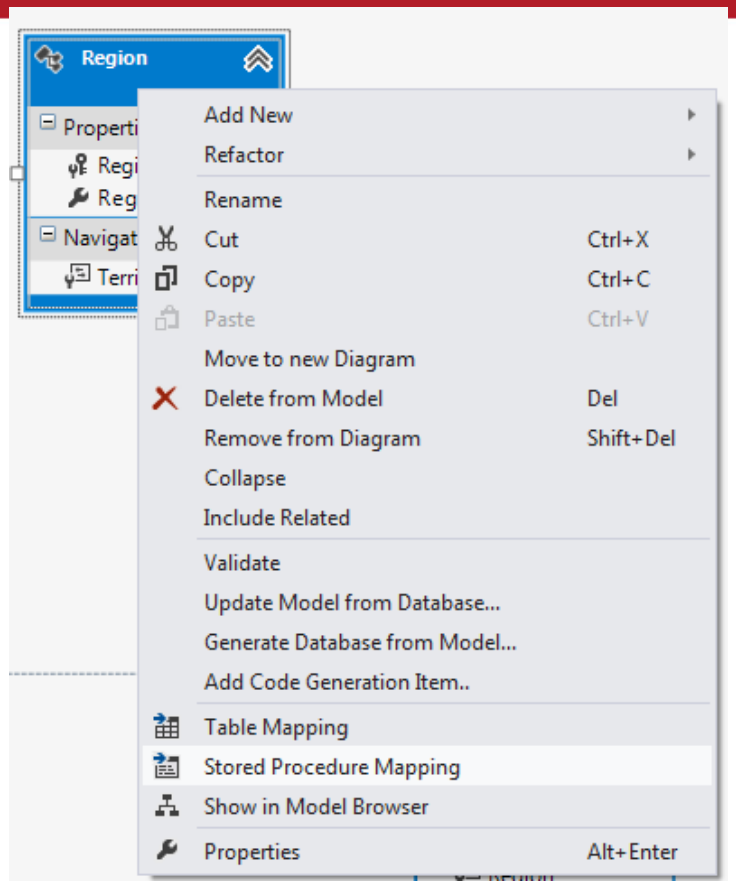
- In many production environments, the database administrators do not allow direct access to the tables for modifying data.
  - Why? Because the database administrator is responsible for the integrity of the data, and they only want applications making changes through approved procedures.
- Entity Framework can fairly easily map to functions and stored procedures.

1. Update Model From Database



SOFTWARE GUILD

2. Right Click the Entity
Choose Stored Procedure Mapping



SOFTWARE GUILD

3. Configure Each Stored Procedure

Package Manager Console | Error List | Output | Mapping Details - Region

Parameter / Column	Operator	Property
Functions		
Insert Using RegionInsert		
Parameters		
RegionDescription : nchar	←	RegionDescription : String
Result Column Bindings		
RegionId	→	RegionID : Int32
<Add Result Binding>		
Update Using RegionUpdate		
Parameters		
RegionID : int	←	RegionID : Int32
RegionDescription : nchar	←	RegionDescription : String
Result Column Bindings		
<Add Result Binding>		
Delete Using RegionDelete		
Parameters		
RegionID : int	←	RegionID : Int32

SOFTWARE GUILD

4. Run for Great Success!

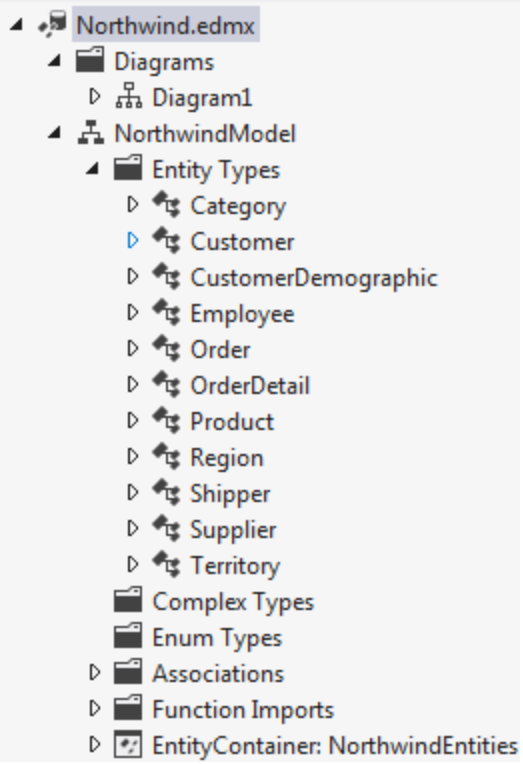
```
private static void RegionTest()
{
    Console.WriteLine("Creating region Test");

    using (var context = new NorthwindEntities())
    {
        Region r = new Region();
        r.RegionDescription = "Test";

        context.Regions.Add(r);
        context.SaveChanges();

        Console.WriteLine("New RegionID: {0}", r.RegionID);

        r.RegionDescription = "Test2";
        context.SaveChanges();

        Console.WriteLine("Check the db");
        Console.ReadLine();

        Console.WriteLine("Delete the new record");
        context.Regions.Remove(r);
        context.SaveChanges();

    }
}
```

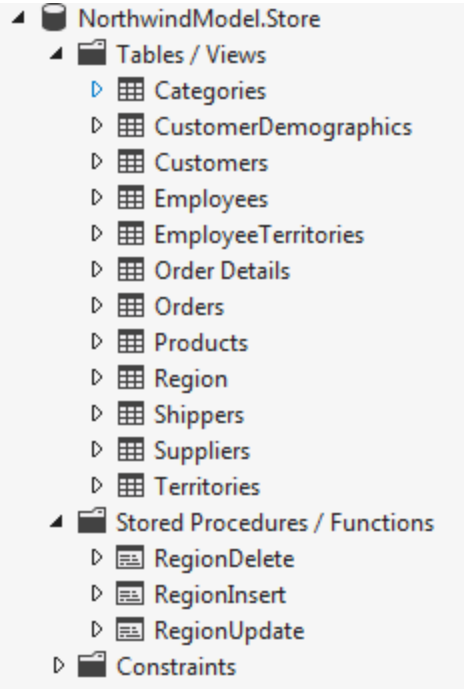
SOFTWARE GUILD

The Model Browser

Right-click on the design surface and choose “Model Browser.”

The Model Browser lets you visually explore both the conceptual model and the SSDL.





Conclusion

We learned how to manually keep our model in sync with database changes. We also learned how to work with our database administrators to allow for stored procedure use to give them fine-grained control over data access.

The most secure way of database access is to not give access to underlying tables directly; instead, do all data changes through stored procedures and all selects against views instead of tables.