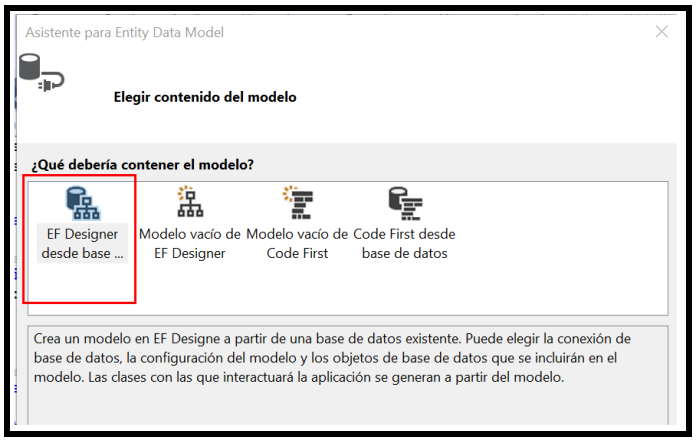
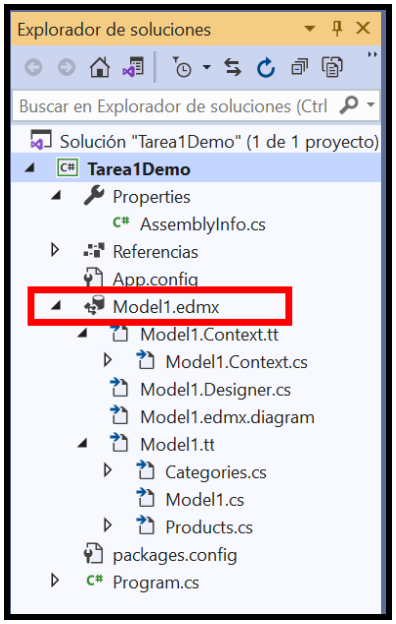
Database First permite realizar ingeniería inversa para obtener **un modelo a partir de una base de datos existente.** El modelo se almacena en un archivo EDMX (extensión **.edmx**) y se puede ver y editar en Entity Framework Designer. Las clases con las que interactuamos en la aplicación se generan automáticamente a partir del archivo EDMX. Además es posible modificar manualmente la base de datos y posteriormente actualizar automáticamente las clases que la mapean. Más info sobre este tema aquí

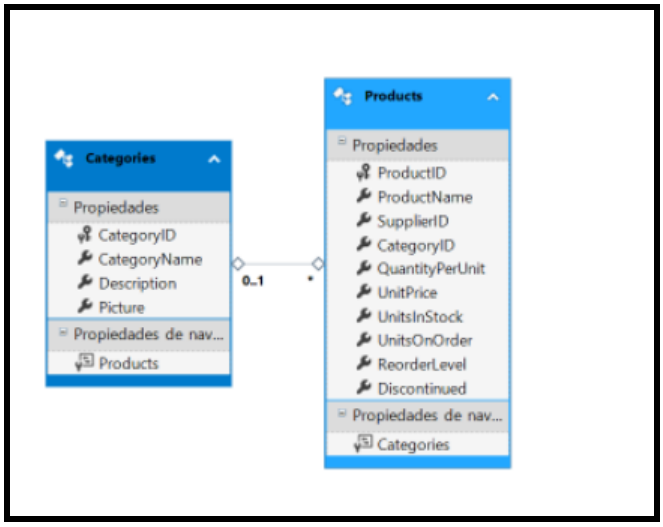
Crear el modelo de datos - Ejemplo
Desde explorador de soluciones / Agregar → Nuevo elemento / seleccionar el modelo de datos de entidad de ADO.NET.



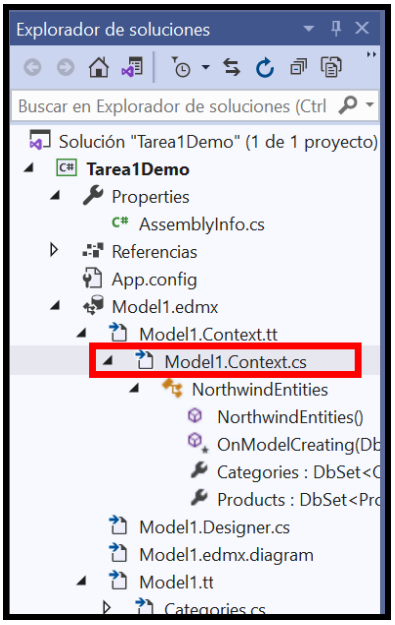
A este asistente le proporcionaremos la información relativa a la cadena de conexión y a las tablas de la base de datos que vamos a mapear, y nos generará a partir de esta información el **modelo de dominio** con el que trabajaremos haciendo uso exclusivamente de nuestro lenguaje de programación C#. Al finalizar el asistente, en el explorador de soluciones encontraremos los siguientes nuevos elementos destacables:

El modelo de datos de entidad *(Model1.edmx)*
Clic en este archivo abre el *diseñador EDM (Entity Data Model)* que muestra todas las entidades de las tablas seleccionadas para el modelo y sus relaciones:





La clase DBContext *(definida en el archivo Model1.Contex.cs)*
Es la clase principal con la que interactúa el modelo. Por defecto su nombre será *NombreBaseDatosEntities* (en el ejemplo ***NorthWindEntities***). Esta clase, junto con el modelo, nos va a permitir escribir y ejecutar consultas, realizar un seguimiento de los cambios realizados en estos objetos, guardar los cambios que hagamos en estos objetos y reflejar estos cambios en la base de datos y vincular objetos a los controles de la interfaz de usuario. Observa cómo los **DBSet** se agregan como propiedades y se asignan a las tablas de la base de datos.

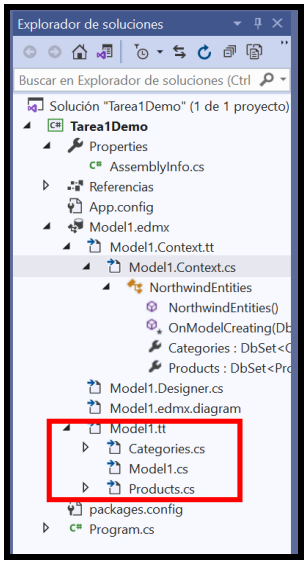



```
1 referencia
public partial class NorthwindEntities : DbContext
{
    0 referencias
    public NorthwindEntities()
        : base("name=NorthwindEntities")
    {
    }

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

    0 referencias
    public virtual DbSet<Categories> Categories { get; set; }
    0 referencias
    public virtual DbSet<Products> Products { get; set; }
}
```

Clases creadas a partir de las tablas *(un archivo .cs por cada tabla importada)*. Estas clases son las que usaremos como modelo al manipular la base de datos.




```
public partial class Categories
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridab
    0 referencias
    public Categories()
    {
        this.Products = new HashSet<Products>();
    }

    0 referencias
    public int CategoryID { get; set; }
    0 referencias
    public string CategoryName { get; set; }
    0 referencias
    public string Description { get; set; }
    0 referencias
    public byte[] Picture { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionProperti
    1 referencia
    public virtual ICollection<Products> Products { get; set; }
}
```

Archivo de configuración (app.config)

Este archivo contiene la cadena de conexión.

```
<connectionStrings>
  <add
    name="NorthwindEntities"
    connectionString="metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provide
    data source=(localdb)\MSSQLLocalDB;initial catalog=Northwind;integrated security=True;Multipl
    providerName="System.Data.EntityClient" />
</connectionStrings>
```

Ahora que tenemos el modelo, podemos usarlo para realizar el acceso a los datos. Las clases que vamos a usar para acceder a los datos son las que se han generado automáticamente en función del archivo EDMX.