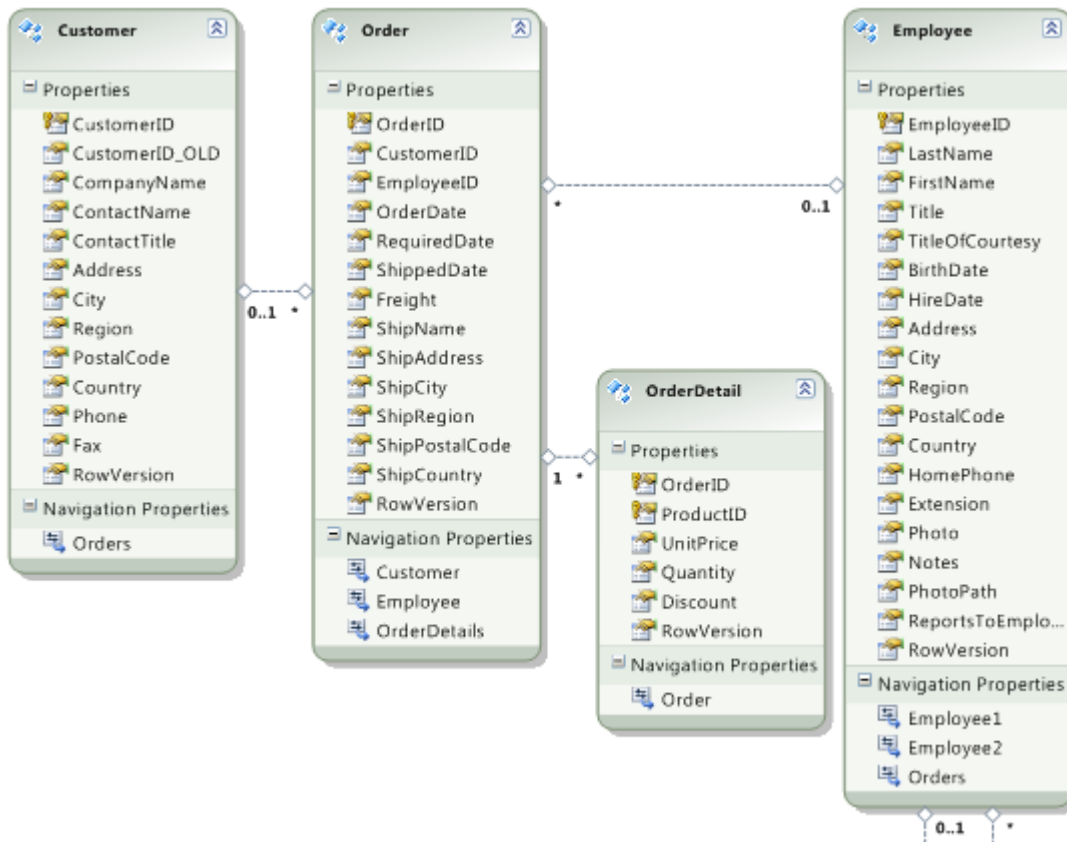


Contents

- [Multi-level undo](#)
- [Actions from the bare canvas](#)
- [Validate the model](#)
- [Entities & associations](#)
- [Renaming](#)
- [Entity property context menu](#)
- [Association context menu](#)
- [Referential Constraint dialog](#)
- [Delete entity](#)

The **Canvas** is the main window in the Visual Studio [EDM Designer](#). It is a design surface that displays a diagram of the conceptual entities that you can navigate and edit. This topic describes some of what you are likely to do on the canvas.

The "canvas" is a visual design surface for editing the conceptual model. You can create, modify, or delete entities and associations by interacting with the diagram graphics.



You should consult [other resources](#) for more a full understanding of how to work on the canvas; [Microsoft's own documentation on the designer](#) is a good place to start.

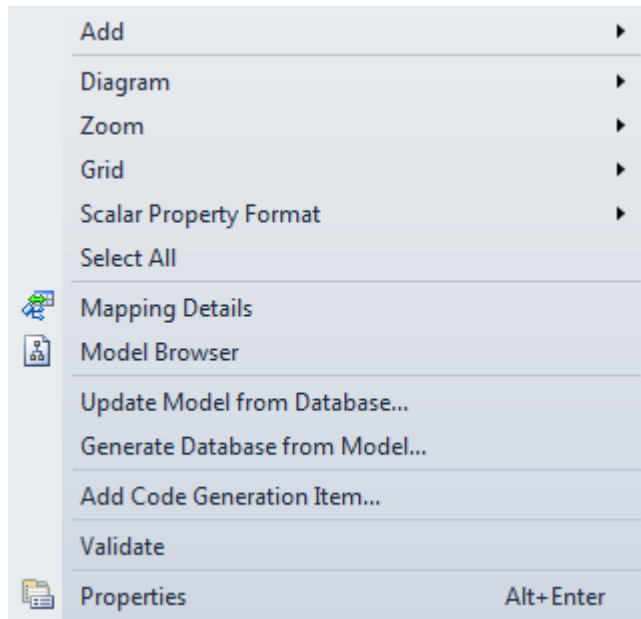
In this topic we mention a few of the most useful gestures and actions, starting with the most important, **undo**.

Multi-level undo

Mistakes are easy to make, especially as you try unfamiliar features such as [deleting entities](#) and adding [ComplexTypes](#). It's a great comfort to know that the designer supports multi-level undo (**Control-Z**) and re-do (**Control-Y**).

Actions from the bare canvas

Right-mouse-click on any blank spot.



This context menu controls the model as a whole.

Add creates a new entity, association, inheritance, or complex type. You're most likely to use this feature in [Model-First development](#); The Data-First developer adds and updates entities via the "Update Model from Database..." option.

Diagram, Zoom, Grid, and Scalar Property Format control the appearance of the diagram.

Model Browser opens the designer's [Model Browser](#) window which is the best way to navigate a large model.

Update Model from Database... is how the Data-First developer adds and updates entities based on database schema. It launches the [Update Model Wizard](#).

Generate Database from Model.. generates SQL scripts for creation of a database that matches the current conceptual model. This is useful for the [Model-First developer](#); the Data-First developer can ignore it.

Properties opens the [designer's Properties Window](#) for the model as a whole.

Validate the model

Validate checks the entire model and reports errors and warnings to the Visual Studio "Error Window". [Microsoft documentation lists some of the errors](#) and suggests remedies.

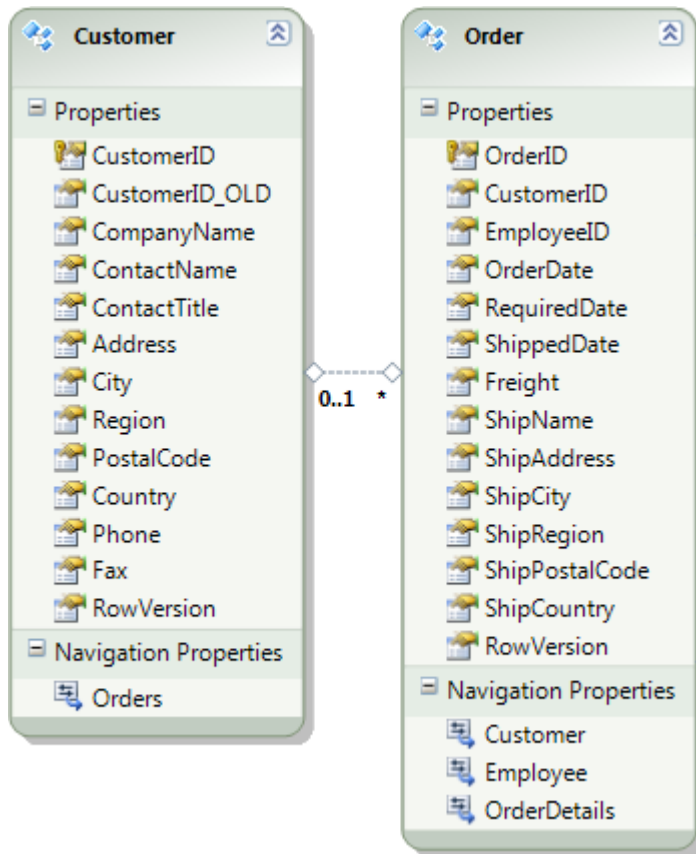
It's worth validating the model regularly while working in the designer, especially after [updating the model from the database](#).

Errors are usually show-stoppers. You won't be able to generate to use the EDM or generate clean entity classes until you fix them.

A few you can safely ignore ... when you understand and accept their causes. In Model-First development you expect to see "Error 11007: Entity type '*SomeEntity*' is not mapped" because you are delaying the mapping exercise until later. It won't prevent code generation.

Entities & associations

This screenshot shows two related **entities**.



The entity names, *Customer* and *Order*, appear in the editable headers at the top of each box.

The line between the entity boxes is the **association** that relates them. The [EDM Wizard](#) produced that association based on the foreign key relationship between the matching "Customer" and "Order" tables. The cardinality of the association is denoted in the diagram by the "0..1 - *" text which means that a *Customer* has zero or more related *Orders*.

The **Properties** section shows both scalar and complex type properties. A **scalar property** returns a single value such as a number or a string. *Customer.CompanyName* is a simple property returning a string. These two entities only have scalar properties.

A **complex type** property [combines multiple inter-related properties into a single type](#). Both *Customer* and *Order* might benefit from combining five of their properties into a common *Address* complex type.

Navigation Properties appear at the bottom. A navigation property traverses an association from one entity to another. The *Customer.Orders* property is a **collection navigation property** that returns child orders of a parent customer. The *Order.Customer* property is a **reference navigation property** that returns the parent customer of a child order. Order has two other navigations to entities not shown.

DevForce implements both reference and collection [navigation properties as specialized entity queries](#).

You **cannot change the display order** of the properties in the diagram from within the designer. The display order is determined by the order of the properties in the EDMX file. You can edit the EDMX file and re-arrange the XML property definitions in the CSDL section if the display order matters to you.

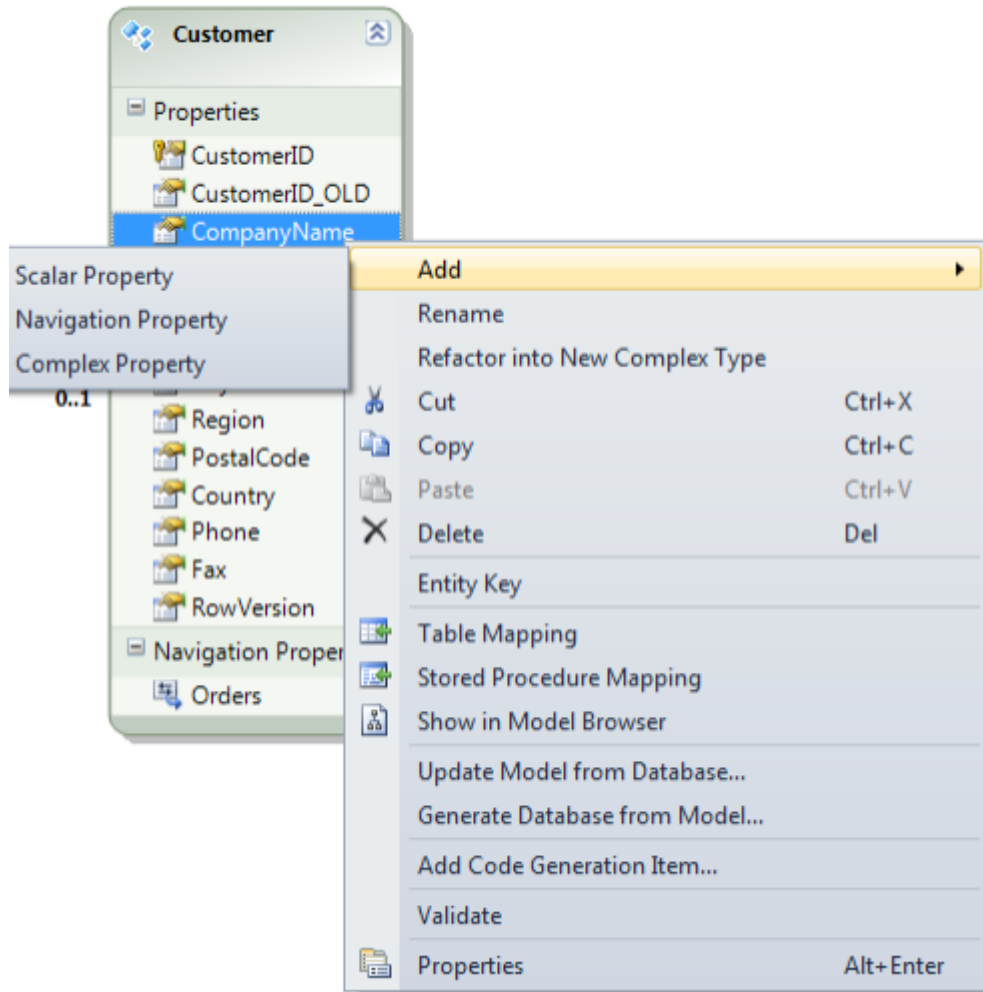
Renaming

You can rename an entity or its property in the object on the diagram.



Entity property context menu

Right-mouse-click on a property to get its **context menu**



Add new properties to the conceptual entity in any of three flavors. You're most likely to use this feature to add a [ComplexType](#) property or when developing in [Model-First style](#). The Data-First developer adds and updates entities via the "Update Model from Database..." option.

Refactor into New Complex Type after selecting multiple properties to be [combined into a ComplexType](#).

Delete removes the property from the entity, leaving the corresponding store column unmapped. You can re-map that column later. The model validates if the store column remains unmapped as long as it is nullable or has a store-determined default value. Validate reports an error otherwise.

Entity Key should be checked if the property participates in the [EntityKey](#); *CompanyName* does not.

Table Mapping opens the [Mapping Window](#) and displays the mappings for the entity to which this property belongs - *Customer* in this case.

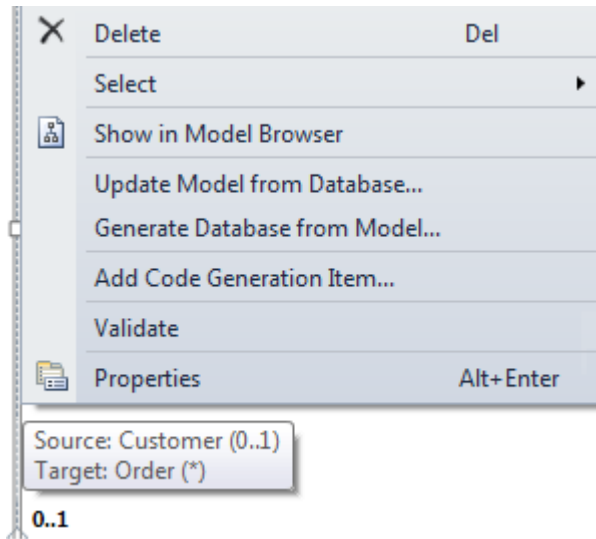
Stored Procedure Mapping opens the [Mapping Window](#) and displays the stored procedure mappings for the entity to which this property belongs. DevForce can take advantage of Entity Framework support for entity stored procedures.

Show in Model Browser opens the [Model Browser window](#) and positions focus on the selected property in the selected entity. The Model Browser is often a more convenient way to get around in a large model. A similar option in the Model Browser takes you back to the property in the diagram.

Properties opens the [designer's Properties Window](#) for the selected property.

Association context menu

Right-mouse-click on the **association** line between two entities to see the **context menu**



Delete removes the association and simultaneously deletes both of its navigation properties. The model will fail to validate if the storage model mapped that association to a foreign key relationship in the database.

Select opens a sub-menu which can take you quickly to either end of the association line. It's enormously helpful when navigating a complex diagram with many long, overlapping association lines.

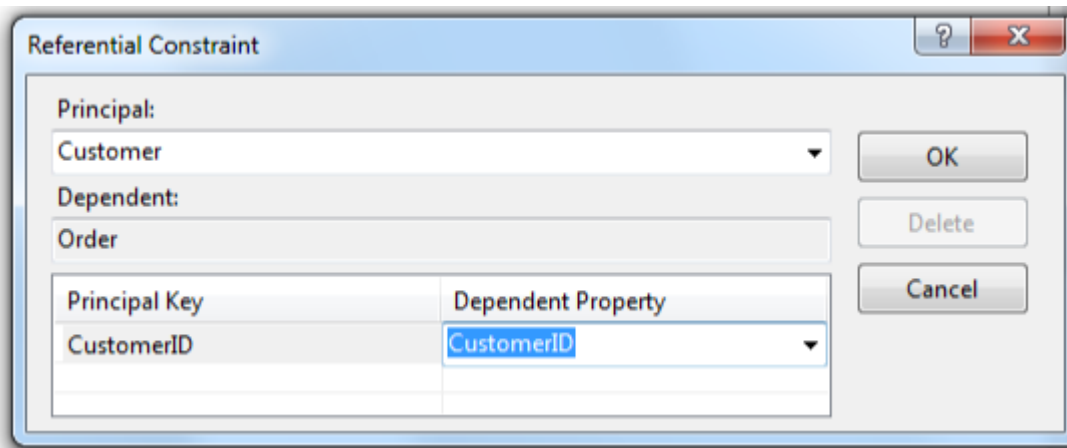
Show in Model Browser opens the [Model Browser window](#) and positions focus on the association. The Model Browser is often a more convenient way to get around in a large model. A similar option in the Model Browser takes you back to this association in the diagram.

Properties opens the [designer's Properties Window](#) for the selected association.

Hover over the association line to reveal the tooltip that indicates (a) the cardinality (0..1 to many) and (b) the source / target (aka, principal / dependent roles) involved.

Referential Constraint dialog

Double-click an association line to reveal the **Referential Constraint** dialog

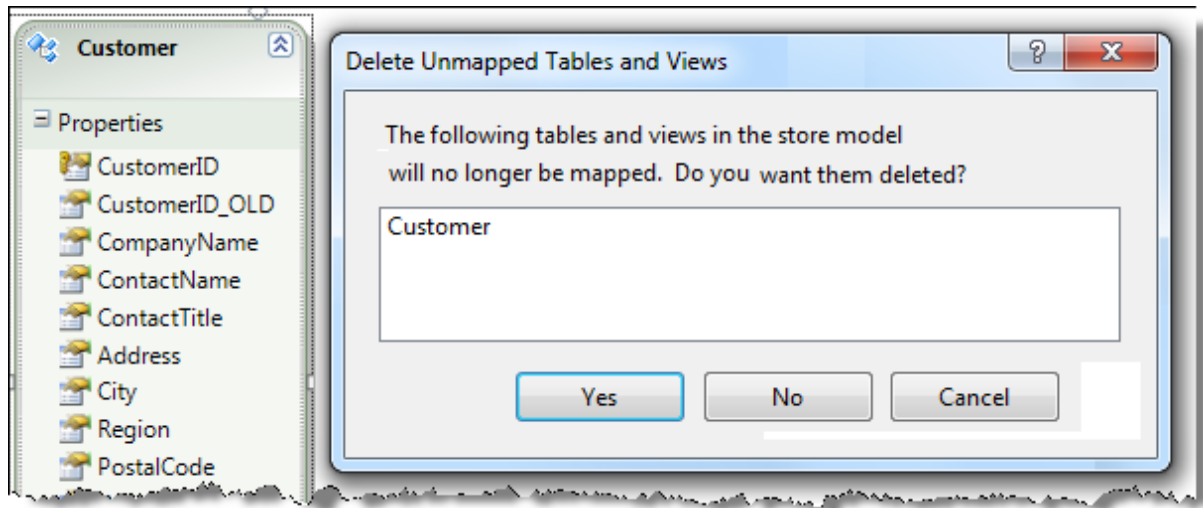


Every association in a DevForce EDM must be a ["Foreign Key" association with referential constraints defined](#).

Delete entity

To delete an unwanted entity, **select it** in the canvas, open its **context menu** and pick **Delete**. The "Delete Unmapped Tables and Views Dialog" appears.

The dialog name is misleading. This is the dialog to delete unwanted entities. It can delete the entity's mapped table (or view) in the process.



The dialog offers three choices which [Microsoft seems to have documented incorrectly](#). The empirical behavior is as follows:

- **Yes:** Deletes both the selected conceptual model objects and the storage model objects to which they were mapped. It is on this point that reality and the Microsoft documentation differ.
- **No:** The selected conceptual model objects will be deleted. "No" means that no storage model objects will be deleted.
- **Cancel:** The operation is canceled. No objects are deleted.

Yes is the default and is usually what you want. You'd pick **No** if you wanted to remove the entity but keep the table. That's rare; you might do this when "splitting an entity" among two tables as shown in [Matthieu Mezil's video](#).

When you say **No**, the entities are gone but their tables remain in the Model. Note that EF doesn't like unmapped tables; the model will fail validation. If you decide not to map the table after all, use the [Model Browser to drop it](#) from the model.