**1:**

**(a)** When you try to delete a Customer, this error throws: 'The DELETE statement conflicted with the REFERENCE constraint "FK\_Orders\_Customers". The conflict occurred in database "Northwind", table "dbo.Orders", column 'CustomerID'.'.

This means that the CustomerID column is being used elsewhere as a foregin key, which inhibits deletion of the row.

If I was to propose a different strategy and was sure the boss wanted all records of this customer gone, it would be to set cascade rules to the foregin key relationship for the UPDATE and DELETE actions.

Otherwise, for reporting purposes, you could add a column and name it DeleteDate column and leave it null until a 'soft' delete occurs where you set the DeletionDate with a datetime to indicate the a record has been deleted.

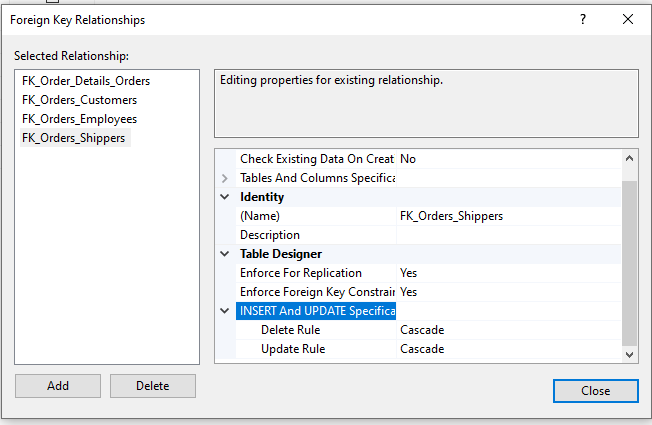
For this example I'm going to stick with the cascade rules and apply the rules to realtionships: **Customers-orders** and **Orders-OrderDetails**.

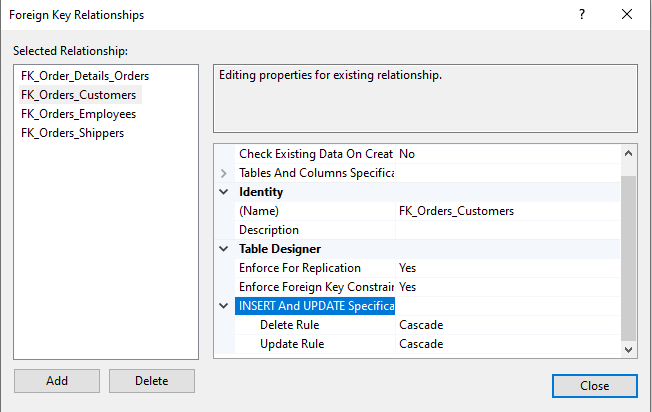
**(b)** When you try to delete a Shipper, this error throws: 'The DELETE statement conflicted with the REFERENCE constraint "FK\_Orders\_Shippers". The conflict occurred in database "Northwind", table "dbo.Orders", column 'ShipVia'.'.

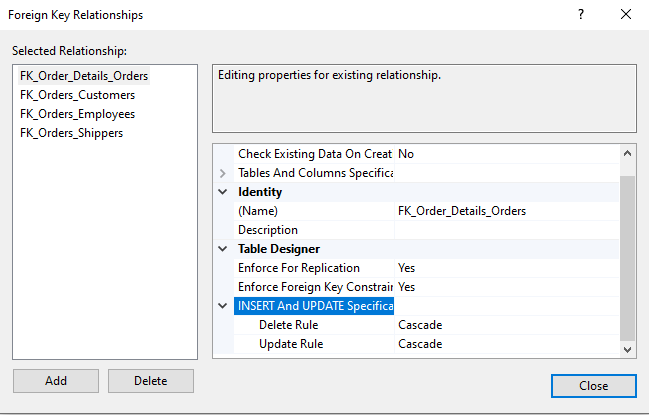
This means there's a foregin key on the Orders table that's using a column in the Shipper.Row to be deleted. My strategy is the same as before (assuming the boss wants this data truncated) and is to set the UPDATE and DELETE relationship rules to CASCADE

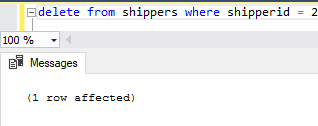
for the relationship: **Orders-Shippers**.

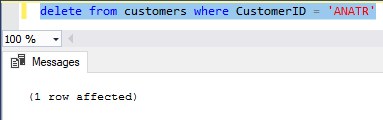
NOTE: When I truncate or delete a Customer or Shipper, for foreign keys that use a column in the row being deleted, the corresponding foregin key row is deleted on the respective table.











Now I’m wishing I made a backup before deleting things… I guess I could always rollback the db. I’m excited to learn how to do that!