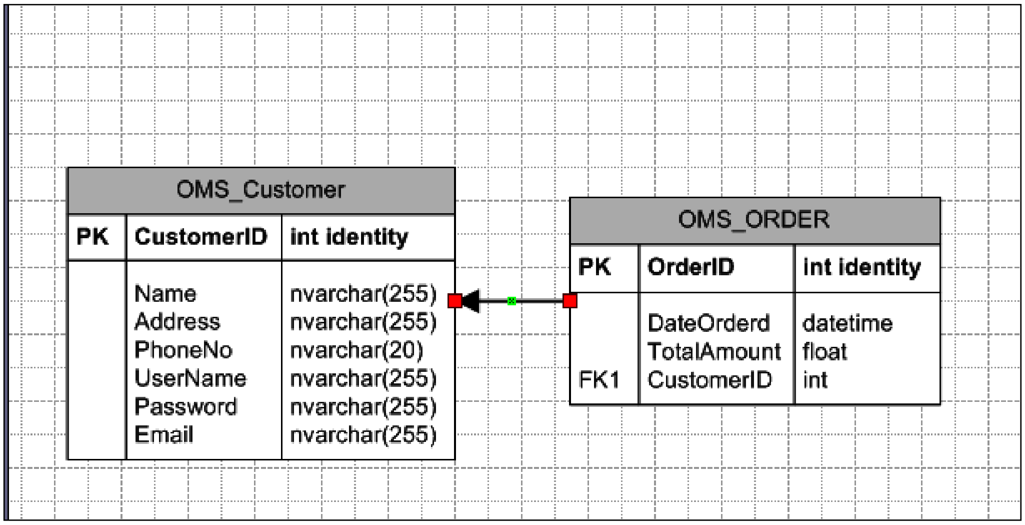


OMS_Customer		
PK	CustomerID	int identity

	Name
	Address
	PhoneNo
	UserName
	Password
	Email
		nvarchar(255)
		nvarchar(255)
		nvarchar(20)
		nvarchar(255)
		nvarchar(255)
		nvarchar(255)

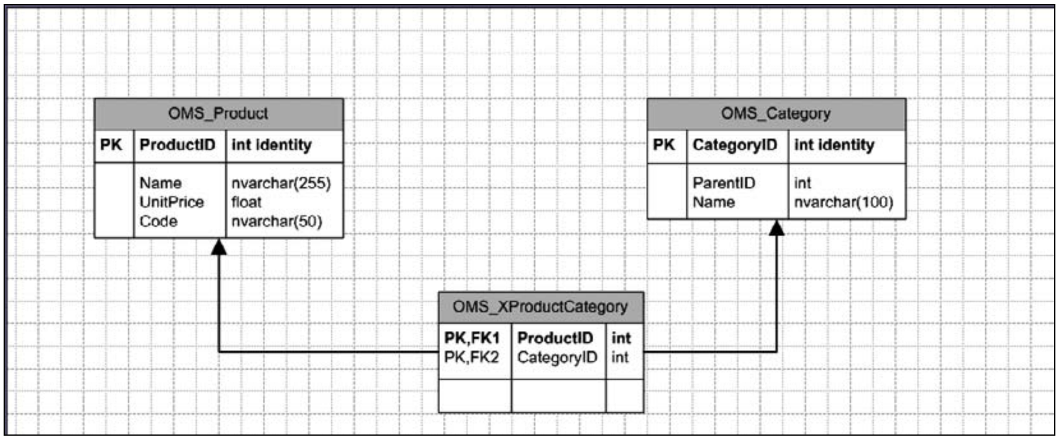
OMS_ORDER		
PK	OrderID	int identity
  FK1	DateOrderd TotalAmount CustomerID	datetime float int

When creating a relationship, always remember that the arrow should point from the dependent entity to the parent entity. In the above case, `Order` is dependent on `Customer`, so we have the relationship diagram from the `Order` table to the `Customer` table.

To create a many-to-many (m:n) relationship, we need to create a separate table that can hold such a relationship. Such tables are sometimes referred to as mapping tables or cross tables.

In our ER diagram, there is a many-to-many relationship between the `Product` and `Category` entities. Each product can belong to multiple categories, so we cannot put `CategoryID` in the `Product table` as an Foreign Key (FK) because then we will be restricting each `Product` row to have only one `Category` (which would be a one-to-many relationship). Similarly, each `Category` can have multiple products listed under it, so we cannot add `ProductID` as a foreign key in the `Category` table, because then we will have a one-to-many relationship between `Category` and `Products`. So to have a many-to-many relationship, we need to create a new table which will contain only the `ProductID` and `CategoryID` columns, so that we can add multiple combinations of `Product-Category` to it. We will do this by creating a table named: `OMS_XProductCategory`. We can use any naming convention here but it is better to follow a certain standard and stick to it.

Here we have used "X" to signify that this table is a "cross" table. Once we have created the table, we will drag and drop two relationship connectors onto our drawing and add relationships from both of the `OMS_Product` and `OMS_Category` tables to the `OMS_XProductCategory` table as shown here:



OMS_Product

PK	ProductID	int identity

	Name
	UnitPrice
	Code
		nvarchar(255)
		float
		nvarchar(50)

OMS_Category

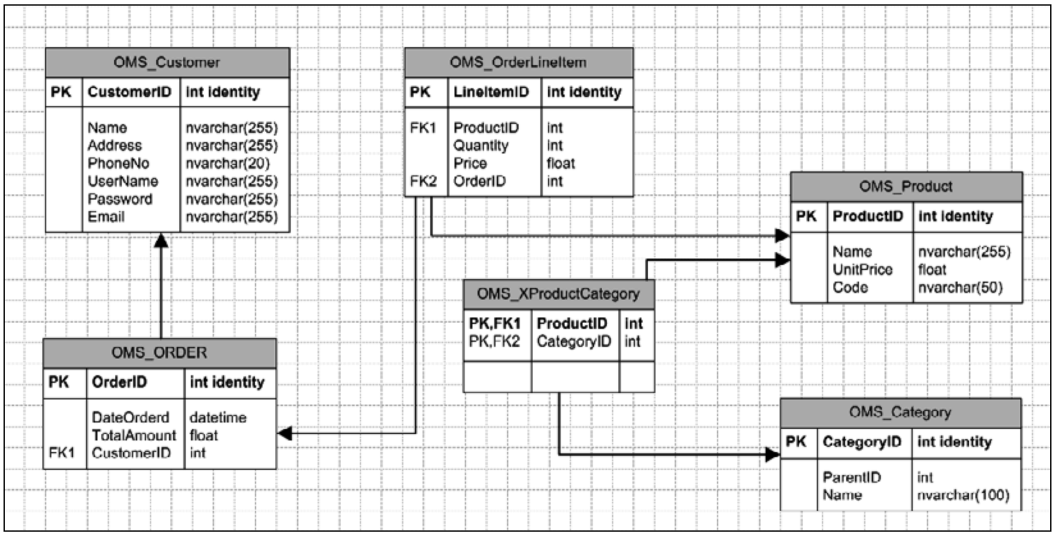
PK	CategoryID	int identity
	ParentID Name	int nvarchar(100)

OMS_XProductCategory

PK,FK1 PK,FK2	ProductID CategoryID	int int

After adding the `ProductId` and `CategoryID`, we mark them as required (by checking **Req'd** in the **Database Properties** box) and set them both as the Primary Key (PK), making the combination of `CategoryID` and `ProductID` a **Composite key** in the `OMS_XProductCategory` table.

Here is the final physical data model, after adding all of the relationships and data types:



OMS_Customer

PK	CustomerID	int identity

	Name
	Address
	PhoneNo
	UserName
	Password
	Email
		nvarchar(255)
		nvarchar(255)
		nvarchar(20)
		nvarchar(255)
		nvarchar(255)
		nvarchar(255)

OMS_OrderLineItem

PK	LineItemID	int identity
FK1 FK2	ProductID Quantity Price OrderID	int int float int

OMS_Product

PK	ProductID	int identity

	Name
	UnitPrice
	Code
		nvarchar(255)
		float
		nvarchar(50)

OMS_XProductCategory

PK,FK1 PK,FK2	ProductID CategoryID	int int

OMS_ORDER

PK	OrderID	int identity
  FK1	DateOrderd TotalAmount CustomerID	datetime float int

OMS_Category

PK	CategoryID	int identity
	ParentID Name	int nvarchar(100)