1NF: FÖRSTA NORMALISERINGSFORMEN

HANDLAR OM ATT VARJE RAD MÅSTE UNIKT KUNNA IDENTIFIERAS
SAMT ATT EN KOLUMN FÅR ENDAST INNEHÅLLA ETT VÄRDE.

CUSTOMERS

Id	Name	Address
1	Hans Bengt Lennart Mattin-Lassei	Nordkapsvägen 1, 123 45 Vega
2	Tommy;Bengt;Lennart;Mattin-Lassei	Smedjegatan 14, 732 30 Arboga
1	Joakim Wahlström	Grusvägen 141, 442 12 Västerås

FEL!
Så här får det inte se ut det blir inte sökbart och skapar mer problem

CUSTOMERS

Id*	FirstName	LastName	MiddleName1	MiddleName2	AddressLine	PostalCode	City
1	Hans	Mattin-Lassei	Bengt	Lennart	Nordkapsvägen 1	12345	Vega
2	Tommy	Mattin-Lassei	Bengt	Lennart	Smedjegatan 14	73010	Arboga
3	Joakim	Wahlström			Grusvägen 141	442 12	Västerås

RÄTT!
Så här ska det se ut med första normaliseringsformen varje värde har Sin egna kolumn. Det gör det sökbart

* Är den unika nyckeln som identifierar varje rad så inga dubbletter kan förekomma

2NF: ANDRA NORMALISERINGSFORMEN

HANDLAR OM ATT DET INTE FÅR FINNAS ÅTERUPPREPANDE KOLUMNER OM SÅDANA FINNS
SÅ MÅSTE DET BRYTAS UT I EGNA TABELLER.

CUSTOMERS

Id*	FirstName	LastName	MiddleName1	MiddleName2	AddressLine	PostalCode	City
1	Hans	Mattin-Lassei	Bengt	Lennart	Nordkapsvägen 1	12345	Vega
2	Tommy	Mattin-Lassei	Bengt	Lennart	Smedjegatan 14	73010	Arboga

FEL!
Så här får det inte se ut tänk om någon har fler än två mellannamn då måste databasen designas om etc.

CUSTOMERS

Id*	FirstName	LastName	AddressLine	PostalCode	City
1	Hans	Mattin-Lassei	Nordkapsvägen 1	12345	Vega
2	Tommy	Mattin-Lassei	Smedjegatan 14	73010	Arboga

MIDDLENAMES

CustomerId*	MiddleName*
1	Bengt
1	Lennart
2	Bengt
2	Lennart

RÄTT!
Så här ska det se ut med andra normaliseringsformen varje återupprepande kolumner har nu fått en egen tabell och i det här fallet så är både CustomerId och MiddleName den unika nyckeln som gör att varje rad blir unik och inga dubbletter kan förekomma.

* Är den unika nyckeln som identifierar varje rad så inga dubbletter kan förekomma

3NF: ANDRA NORMALISERINGSFORMEN

HANDLAR OM ATT ALLT MÅSTE VARA HÄRLETT DEN PRIMÄRA NYCKELN, VILKET INNEBÄR ATT ALLT SOM INTE KAN ”IDENTIFIERA” OBJEKTET I TABELLEN SKA BRYTAS UT.

CUSTOMERS

Id*	FirstName	LastName	AddressLine	PostalCode	City
1	Hans	Mattin-Lassei	Nordkapsvägen 1	12345	Vega
2	Tommy	Mattin-Lassei	Smedjegatan 14	73010	Arboga

FEL!
Så här får det inte se ut tänk om någon ropar Arboga, det kan vara flera personer Så adressinformation ska brytas ut.

CUSTOMERS

Id*	FirstName	LastName	AddressId
1	Hans	Mattin-Lassei	1
2	Tommy	Mattin-Lassei	2

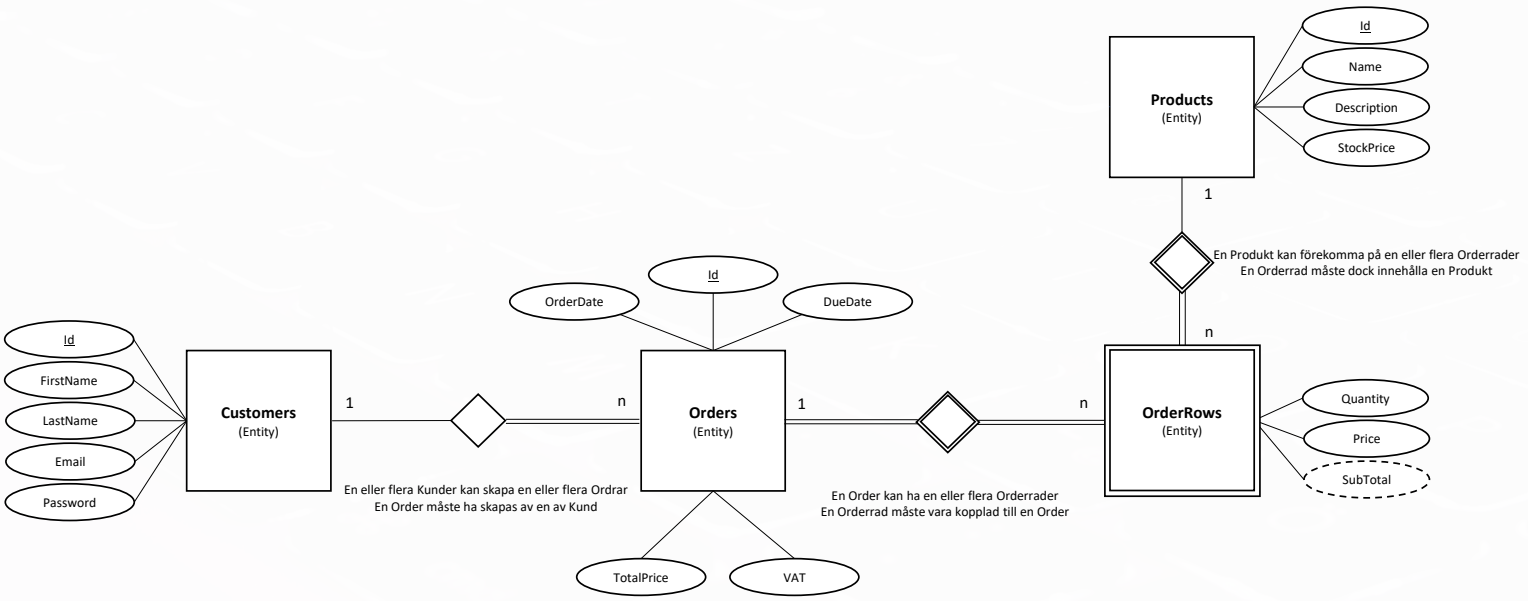
ADDRESSES

Id*	AddressLine	PostalCode	City
1	Nordkapsvägen 1	12345	Vega
2	Smedjegatan 14	73010	Arboga

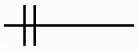
RÄTT!
Så här ska det se ut med tredje Normaliseringsformen. All adressinformation har fått en egen tabell och sedan länkas informationen in med hjälp av en främmande nyckel (FK) i Customers. Om Det skulle vara så att man har flera adresser så gör man det som gjordes i 2NF.

* Är den unika nyckeln som identifierar varje rad så inga dubbletter kan förekomma

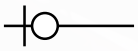
ITERATION #2



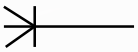
RELATIONER/CONSTRAINTS



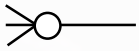
Denna relation säger en och endast en



Denna relation säger noll och till en

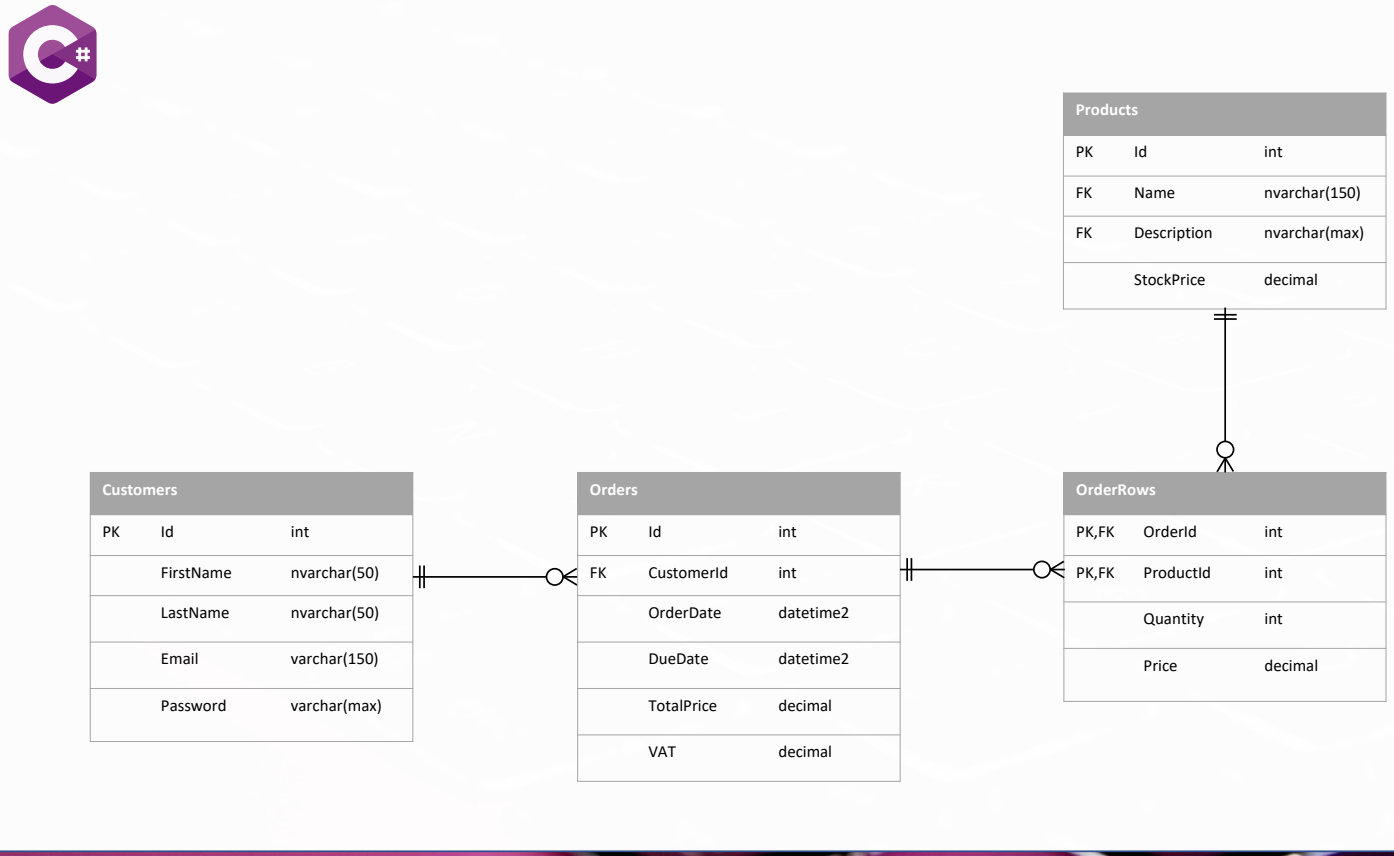


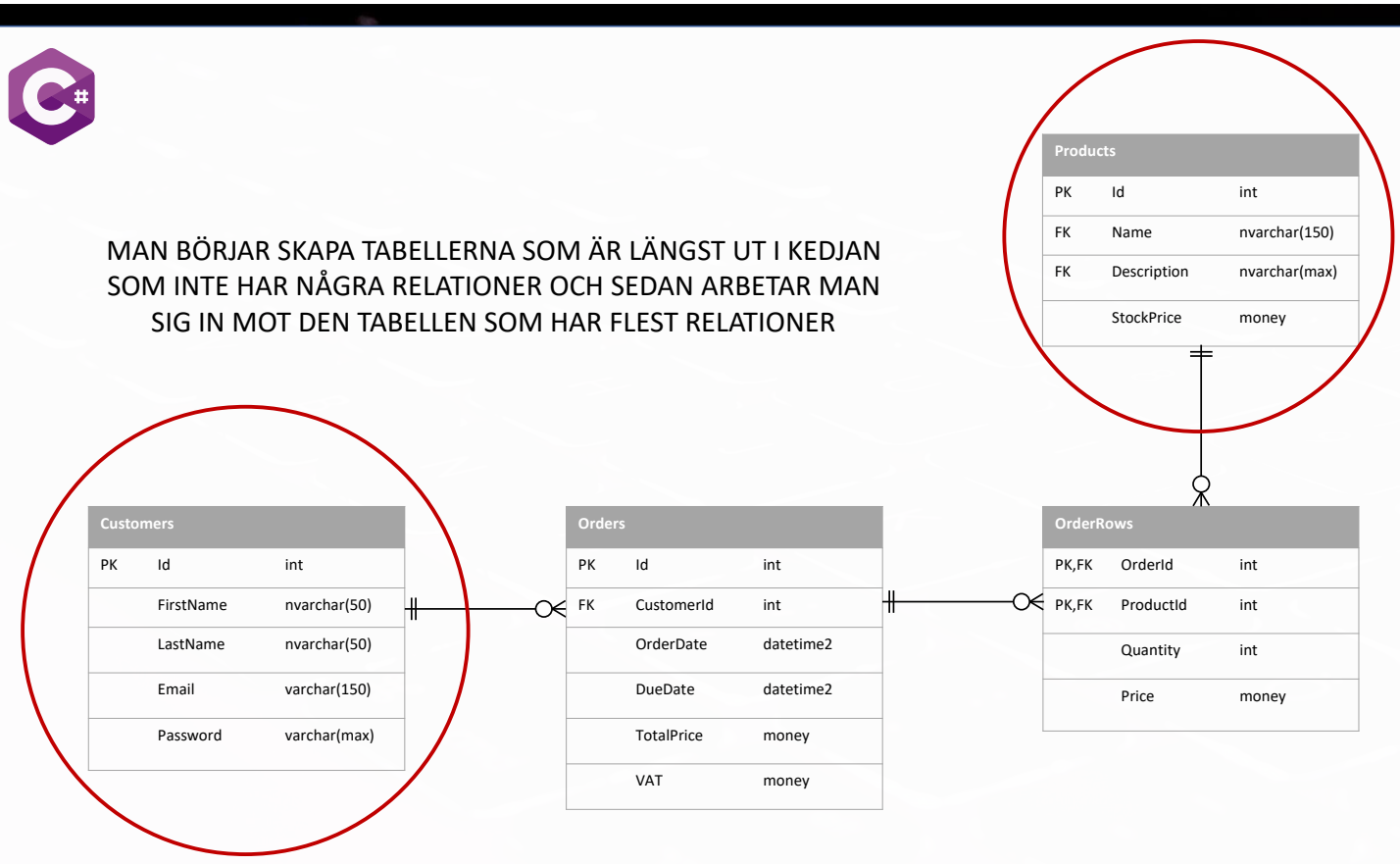
Denna relation säger en och till många



Denna relation säger noll och till många

C#	SQL	Storlek	Beskrivning
String	char(10)	1 byte	fast storlek, statisk, snabb
	nchar(10)	2 byte	unicode
	varchar(10)	1 byte + 2 byte	dynamisk storlek, långsammare
	nvarchar(10)	2 byte + 2 byte	unicode
Int	tinyint	0 till 255	
	smallint	-32768 till 32767	
	int	-2147483648 till 2147483647	
	bigint	-9223372036854775808 till 9223372036854775807	
Guid	uniqueidentifier		
DateTime	datetime, datetime2, datetimeoffset		
	bit, money, float, date, datetime, datetime2, numric, decimal, real, time, smalldatetime, datetimeoffset, text, ntext, binary, varbinary, image, uniqueidentifier		


Products
PK Id int
FK Name nvarchar(150)
FK Description nvarchar(max)
StockPrice decimal
Customers
PK Id int
FirstName nvarchar(50)
LastName nvarchar(50)
Email varchar(150)
Password varchar(max)
Orders
PK Id int
FK CustomerId int
OrderDate datetime2
DueDate datetime2
TotalPrice decimal
VAT decimal
OrderRows
PK,FK OrderId int
PK,FK ProductId int
Quantity int
Price decimal


MAN BÖRJAR SKAPA TABELLERNA SOM ÄR LÄNGST UT I KEDJAN SOM INTE HAR NÅGRA RELATIONER OCH SEDAN ARBETAR MAN SIG IN MOT DEN TABELLEN SOM HAR FLEST RELATIONER
Products
PK Id int
FK Name nvarchar(150)
FK Description nvarchar(max)
StockPrice money
Customers
PK Id int
FirstName nvarchar(50)
LastName nvarchar(50)
Email varchar(150)
Password varchar(max)
Orders
PK Id int
FK CustomerId int
OrderDate datetime2
DueDate datetime2
TotalPrice money
VAT money
OrderRows
PK,FK OrderId int
PK,FK ProductId int
Quantity int
Price money

Customers		
PK	Id	int
	FirstName	nvarchar(50)
	LastName	nvarchar(50)
	Email	varchar(150)
	Password	varchar(max)

```
CREATE TABLE Customers (
    Id int not null identity primary key,
    FirstName nvarchar(50) not null,
    LastName nvarchar(50) not null,
    Email varchar(150) not null unique,
    Password varchar(max) not null
)
GO
```

not null = talar om att man måste fylla i detta fält för att informationen ska sparas i databasen
identity = stegar automatiskt upp ett värde (måste vara tinyint, smallint, int, bigint)
primary key = talar om att det här värdet ska vara den unika nyckeln i tabellen så att inte dubbletter kan förekomma
unique = gör värdet till unikt och det gör att ingen annan rad kan ha samma värde.
GO = talar om för kompilatorn att nu ska du göra det som står här ovanför, sedan fortsätta läsa filen