DB-modellering

Dokumentation byggd på lektion #9 i kursen

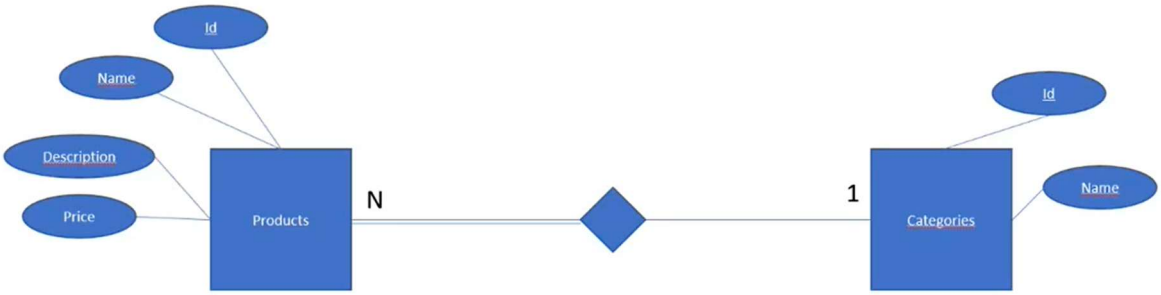
Sammanfattning:

För att skapa ett projekt av C# , SQL och ev UI börjar man med att ta reda på förutsättningarna genom att skapa ett en bild (ERD) som förutsättningslöst beskriver den datastruktur som ska till. Vad för data och hur är de relaterade till varandra?
Man kan kombinera med nästa steg, men då riskerar man lätt att fastna i tekniska problem i st. f. att modellera en korrekt struktur.

Sedan modelleras ovanstående med de förutsättningar som ditt val av databas medför (t.ex. SQL) i ett 'Crow foot'-diagram.

Med detta underlag kan man omsätta arbetet till kodning av SQL och c# enligt fasta principer, bl.a. med testning av SQL-kommandon i terminal *innan* inkopiering i c#-koden (Då det är mycket svårare att inse fel och brister i sql-kod där).

ER Diagram (förutsättningslös modell):



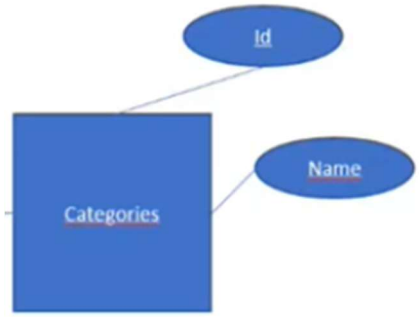
Entitet (Motsv. klass i c#)

Relation.
=== Tvingande relation (En produkt måste ha en kategori)
---- Ej tvingande relation (En kategori kan vara oanvänd)

Attribut (motsv. property i c#)

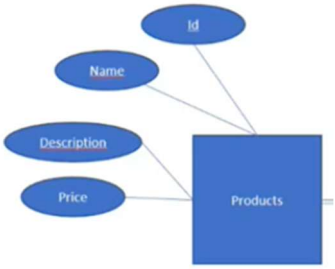
ER modell (Nu börjar vi sätta MS-SQL tänk:

Tabeller



Categories			
PK	Id	int	Not Null
U	Name	nVarChar(50)	Not Null

n (i nVarChar) innebär unicode-stöd
U = Unique



Products			
PK	Id	int	Not null
U	Name	nVarChar(150)	Not null
	Description	nVarChar(max)	
	Price	money	Not null
FK	CategoryId	int	Not null

nVarChar(max) = 2 Gb

Crow foot diagram:

blir nu en linje mellan tabellerna med “kråkfötter”

Products			
PK	Id	int	Not null
U	Name	nVarChar(150)	Not null
	Description	nVarChar(max)	
	Price	money	Not null
FK	CategoryId	int	Not null

Categories			
PK	Id	int	Not Null
U	Name	nVarChar(50)	Not Null

Noll, en eller flera relation. En Kategori kan t.o.m. finnas utan produkt.

En strikt relation. En produkt måste ha en (och bara en) kategori.

Nästa steg är att skapa en databasfil i vårt projekt:

- Skapa en mapp i projektet och döp den till “Data”
- Kör: Server Explorer - Data Connection - Add Connection
- Välj: Microsoft SQL-server database file
- Tryck browse och manövrera till din nyss skapade data-mapp
- Ge ett filnamn (t.ex. sql_db) och tryck OK för att skapa.
- Högerklicka på din nyskapade db och välj New Query

SQL-kommandon:

Först droppar vi tabellnamnen, ifall vi kört dem tidigare, så vi får fräsch start. Sedan skapar vi dem efter vår modell ovan:

```
DROP TABLE Products
DROP TABLE Categories

CREATE TABLE Categories(
Id int not null identity primary key,
Name nvarchar(50) not null unique
)

GO

CREATE TABLE Products (
Id int not null identity primary key,
Name nvarchar (150) not null unique,
Description nvarchar(max) null,
Price money not null,
CategoryId int not null references Categories (Id)
)
```

‘GO’ mellan ‘CREATE’-satserna gör att Categories säkert har skapats innan Products blir till.

identity innebär att indexet skapas och automatiskt räknas upp för varje ny rad.

C#:

Skapa klasser:

Modeller:

Skapa en mapp: Models
Sedan skapar du en modellklass för produkt - t.ex. med följande ‘fyllning’:

```
public int Id { get; set; }
public string Name { get; set; } = null! ;
public string Description { get; set; } = null! ;
public decimal Price { get; set; } = 0;
public int CategoryId { get; set; }

public Category Category { get; set; } = new Category()
```

Sedan kan man skapa Category-modellen genom att högerklicka på den röd-understrukna Category och välja ‘Quick action ..’ - ‘Generate type …’ - ‘ … new file’

Öppna den nya klassen och fyll i:

```
public int Id { get; set; }
public string Name { get; set; } = null!;
```

Services:

Som nästa steg behövs kod för att hantera vår data. Börja med att skapa en mapp: Services

I den skapas en klass, CategoryService som i Hans demo började med följande:

```
private readonly string _connectionstring = “”
public void Create(string name) {}

public Category Get(int id) {}

public IEnumerable<Category> GetAll() {}
```

Högerklicka på databasen (i server explorer) och välj properties. Kopiera sedan ‘connection’-värdet till ‘_connectionstring’ i filen (glöm inte @ !):

```
private readonly string _connectionstring = @”Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename="C:\C-Sharp-
fundamentals\cs\Lektion 9\SQL-
Example\Data\sql_example_db.mdf";Integrated Security=True;Connect
Timeout=30”
```

Sedan följer ifyllning av metoden ‘Create’ - Skriv följande:

```
using (SqlConnection)
```

Vid röd understrykning: Välj ‘Quick action …’ och installera System.Data.SqlClient
Sedan knappar man på …

```
using (SqlConnection conn = new SqlConnection(_connectionstring))
{
      conn . Open () ;
      using (SqlCommand cmd = new SqlCommand(""), conn)
      {
            [koden hit, se nästa avsnitt]
      }
}
```

För att skapa koden som behövs ovan starta en ny **sql-terminal** och provkör:

```
DECLARE @CategoryName nvarchar(50 SET @CategoryName = 'Datorer'

IF NOT EXISTS (SELECT Id FROM Categories WHERE Name =
@CategoryName) INSERT INTO Categories VALUES (@CategoryName)
```

Man kan provköra resultatet med:

```
SELECT * FROM Categories
```

DECLARE-raden är bara för att ha data att provköra med.

Kopiera If-satsen och klistra in mellan “” i *using (SqlCommand …* ovan

Sist fyller vi ‘kroppen’ på Create-metoden:

```
cmd.Parameters.AddWithValue("@CategoryName", name) ;
cmd.ExecuteNonQuery() ;
```

Första raden kopplar ihop c#-variabeln med SQL-ditto.
Andra raden kör kommandot utan att fånga upp något svar.

Allt annat följer ovanstående arbetssätt, med det finns såklart genvägar och skillnader i kommandosyntax som kommer i nästa uppdatering av detta dokument!

Tack för att du läste och hoppas det gör någon nytta i dina ansträngningar att lära.
Speciellt som det tar fyra timmar att producera en halvtimme videolektion …