# Beskrivelse af SQL

Vores database består af 6 forskellige tabeller. Category, CustomerTable, Employee, Order, OrderLine og Product.

I vores projekt er der kun forbindelse til CustomerTable og Product. De andre er taget med for at illustrerer vores ER diagram og fordi de er forbundet til hinanden med primær- og fremmednøgler. Eksempel på SQL-kode, vi har benyttet til at oprette tabellerne:

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, nummer/tal

Automatisk genereret beskrivelse

Bemærk at vi har sat systemet til selv at tildele værdien til CustomerID. Det har vi gjort ved at sætte IDENTITY(1,1) på. Det betyder at den første kunde der bliver oprettet, får CustomerID = 1. og den næste 2 osv. Da den øger med 1 hver gang. Derudover er CustomerID sat til at være primær nøgle.

Alle værdierne er sat til at være NOT NULL hvilket betyder at de skal indtastes for at oprette kunden.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, algebra

Automatisk genereret beskrivelse

Her gør det samme sig gældende med ProductID, det bliver også automatisk oprettet. Samt det er også primær nøgle.

Som noget anderledes har denne en fremmednøgle – hvilket betyder at den er forbundet til tabellen Category med dens primærnøgle CategoryID. Dette for, på sigt, at kunne koble de forskellige varer på forskellige grupper. Dette er desuden et krav fra kunden.

For at sætte varer ind i tabellen benytter vi følgende metode i vores program:

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede

Automatisk genereret beskrivelse

Metoden tager en parameter med som egentligt henviser til vores Models mappe – men i stedet for at der står Models.Product product alle steder den bruges, har vi taget den med i toppen, hvor der i stedet står using ButiksSystem.Models.

Bemærk at metoden ikke selv giver en værdi til ProductID. Værdierne får metoden længere oppe i programkode. De bliver gemt i UI laget og taget med ned igennem Servicelaget. Hvor de til sidst kommer ned i Repositorylaget, som er der denne metode ligger.

Metoden tager input fra tekstbokse, fra brugergrænsefladen, og gemmer disse input i product, og denne bliver taget med ned til repository, og er den der bliver brugt i metoden.

Vores program kan desuden også opdatere informationerne på varer og kunder. Dette gøres også med SQL og metoden ligger også i repository.

Den ser således ud:

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede

Automatisk genereret beskrivelse

Igen er ProductID ikke en variabel der kan redigeres i, men i stedet den programmet pejler efter. Denne metode fungerer på samme facon som CreateProduct. Den tager input fra UI i tekstbokse og gemmer disse info i product, som bliver taget med ned til repository.

Metoden er lavet sådan at alle informationerne om varen bliver automatisk skrevet i tekstboksen i UI-laget, og så kan brugeren ændre i de ønskede felter. Når brugeren trykker på gem, gemmer programmet alle de informationer der ligger i tekstboksene og tager alle disse informationer med videre – på denne måde opdaterer databasen varen med samtlige informationer, også selvom nogen af dem evt. er de samme som før.

I vores projekt og system har vi benyttet os af dummydata for at illustrerer et .txt-fil udtræk fra programmet. Denne fil viser en ordreoversigt baseret på valgt dato, og ikke på den enkelte kunde.