## Valg til Entiteterne i databasen:

I Pristjek220’s database er der fire forskellige entiteter, og de hedder Store, Product, HasA og Login. Mellem entiteterne Store og Product er der en mange-til-mange relation, da én forretning kan sælge mange produkter, og ét produkt kan blive solgt i mange forretninger. Denne relation bliver normalt selv oprettet, hvis relationen ikke har nogle andre properties. Da en forretning ikke nødvendigvis sælger et produkt til den samme pris som i andre forretninger, var det i Pristjek220 nødvendigt at have en property til produktets pris på relationen mellem forretningen og produktet. Denne property skal indeholde, hvad prisen for produktet er i lige præcis den forretning, det tilhører.

Løsningen på dette problem blev erfaret gennem undervisningen i DAB[[1]](#footnote-1). Her blev der fundet frem til, at en mange-til-mange relation, hvor der er brug for properties på relationen, skal have oprettet en entitet til relationen. Af denne grund blev entiteten HasA oprettet i Pristjek220. HasA har derfor en property til prisen, samt en relation til en forretning og et produkt, for at binde de to entiteter sammen. Derved kan Pristjek220 håndtere, at en ny forretning åbner og oprette HasA entiteter til de produkter, som er i den nye forretnings sortiment. Ligeledes, hvis en forretning får et nyt produkt i deres sortiment, vil der blive oprettet en HasA entitet mellem forretningen og produktet. Denne håndtering gør samtidig, at der kun skal være én udgave af hvert produkt og af hver forretning, da der kan bindes mange HasA entiteter mellem dem. Modellen for databasen kan ses på Figur 1.



Figur 1 UML diagram for databasen.

Tabel 1: Første udkast af opsætningen af databasen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Aldi | Fakta | Føtex | Kiwi | Rema1000 |
| Banan | 2.95 | 1.95 | 2.50 | 2.55 | 2.70 |
| Tomat | 2.90 | 2.40. | 1.95 | 2.60 | 2.30 |

Inden denne erfaring var planen med den første udgave af databasen, at den skulle indeholde en enkel tabel, hvor man kunne se produktets navn i rækkerne og de forskellige forretninger hen ad kolonerne, og på den måde finde prisen for produktet i den enkelte forretning som kan ses på Tabel 1. Dette viste sig at have nogle problemer, da Product klassen skulle laves med en variabel for hver forretning, som prisen kunne gemmes i. Det resulterede i, at databasen ikke ville være åben for udvidelser, hvilket er ineffektivt, når forretninger kan åbne og lukke, og forretningerne får nye produkter i deres sortiment.

For at forretningerne ikke kan ændre prisen på et produkt i en anden forretning, eller ændre på andre forretningers sortiment, er der lavet et login til hver forretning, og alle logins gemmes i databasen. Denne entitet har et brugernavn, kodeord, og en reference til den forretning, den har kontrol over. Når der logges ind for en forretningsmanager, har han derved kun mulighed for at ændre, hvordan hans egen forretning skal fremstå. Derudover er der lavet så Administratoren, som kan tilføje og slette forretninger, også har et login. Dette login er specielt i forhold til de andre, da det giver adgang til administrationsdelen af forretningerne. Derfor er Administratorens login blevet lavet med en forretning, der hedder Admin, og på den måde tages der i programmet højde for, hvilken retning administrationsprogrammet skal tage efter login.

1. I4DAB - Databaser [↑](#footnote-ref-1)