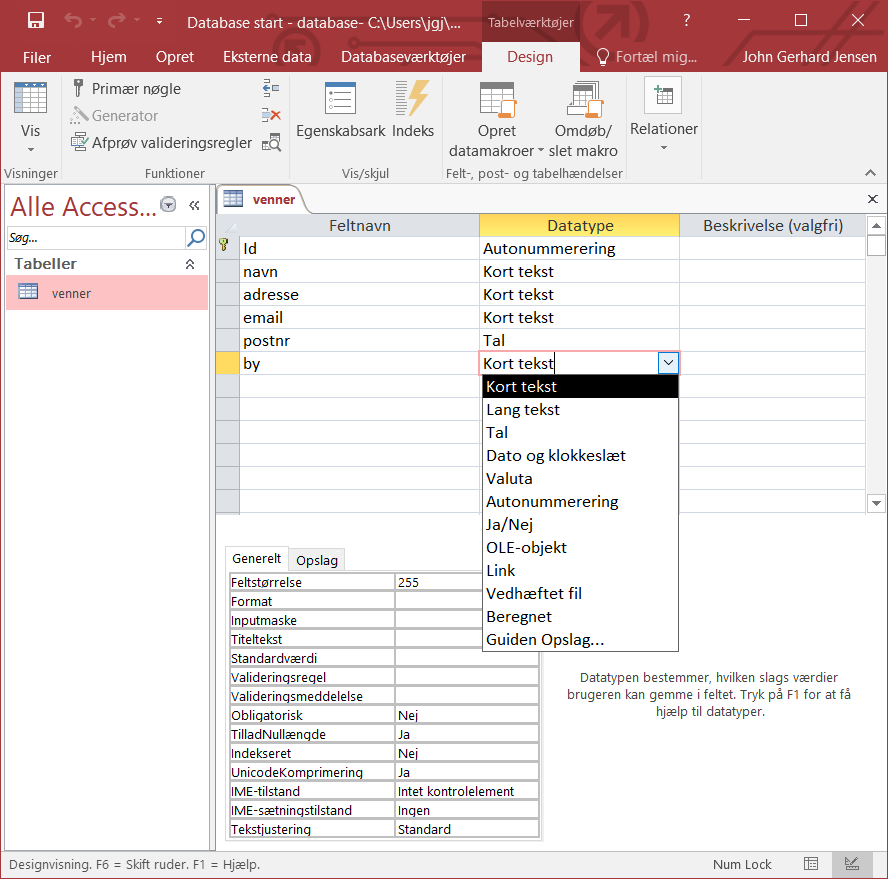
Database intro

Materiale til introduktion af relations databaser på DDU, HC Ørsted, Lyngby 2020-21.

Ord markeret med fed er eksamenspensum, som eleven skal kunne forklare til eksamen

Databaser bruges til at gemme data, især data som man forventer at skal genbruge data. Lectio er et godt eksempel på store mængder af data som skal bruges til registrering af forskellige data og som skal vedligeholdes ofte (dvs dagligt eller flere gange dagligt).

Data gemmes i **tabeller**, der består af rækker af **datafelter** med hvert sit **feltnavn**, **datatype** og **beskrivelse**.



I vinduet fra Access kan man se de forskellige datatyper, som er nogenlunde selvforklarende.

**Autonummerrering** er en tilpasset heltals-tæller (int) som sørger for at en ny datarække får sit eget unikke tal. Det betyder, at hvis man sletter en post (f.eks nr 1) så kommer post nr 1 aldrig igen.

Kort tekst (255 tegn) og lang tekst (4096) er forskellige i størrelse (antal bit), til forskellige længder af tekst.

Ja/Nej er en boolean (true/false)

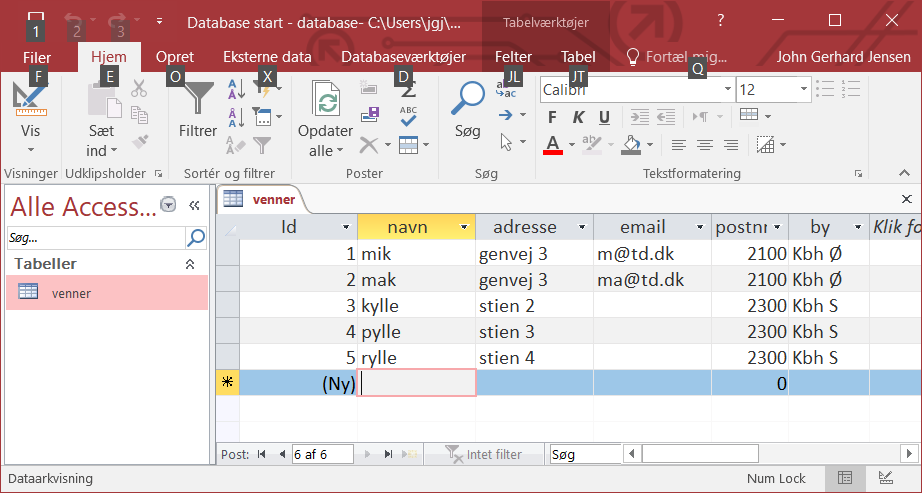
OLE-objekter er hvis man registrerer Objekter (feks fra egne klasser), det er ikke noget vi kommer mere ind på.

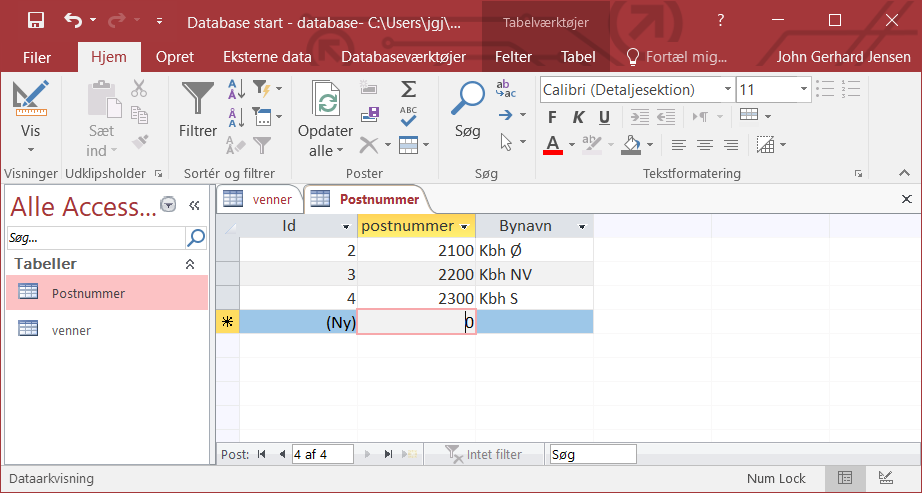
Vedhæftet fil henviser til et sted på HDD, hvor der kan ligge andre filer, det kan være billeder, video, lyd og den slags.

Databaser kan ikke lide redundans, så det vil vi forsørge at undgå, hvis det er muligt.

**Redundans** er gentagelse af data, så hvis vi har designet data der skal tastes flere forskellige steder, har vi lavet et uhensigtsmæssigt design og skal rette det til. (Lidt lige som programmeringskode).

I tabellen herunder er der lavet redundans, da koblingen mellem postnummer og by optræder flere gange. Problemet løses ved at oprette en ny tabel med de redundante data.





Forbindelsen mellem tabeller redder **relationer** – der findes **3 typer relationer:**

**1: 1 en-til-en,** feks person – personnummer. Kender du en så kan du finde den anden

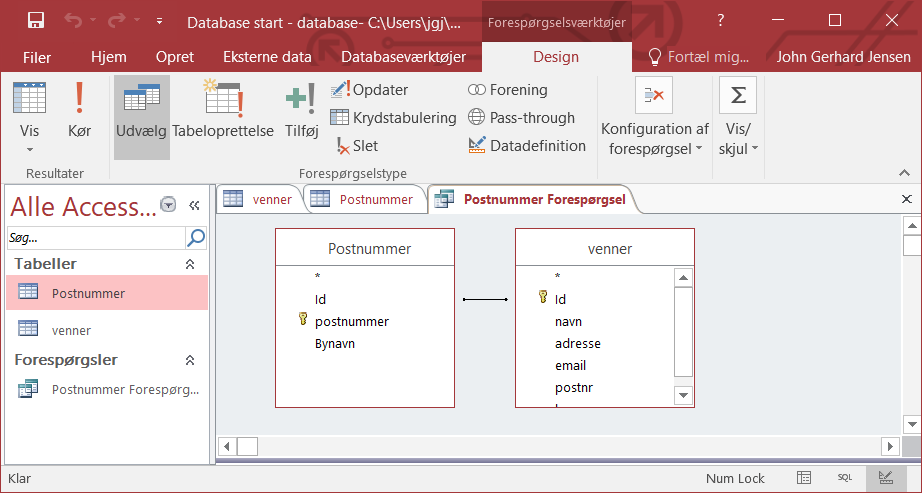
**1:m en-til-mange,** feks person – postnummer. Hver person har et postnummer, der kan være flere med samme postnummer.

**M:N mange til mange,** feks elever – lærere. Hver lærer har mange elever og hver elev har mange lærere.

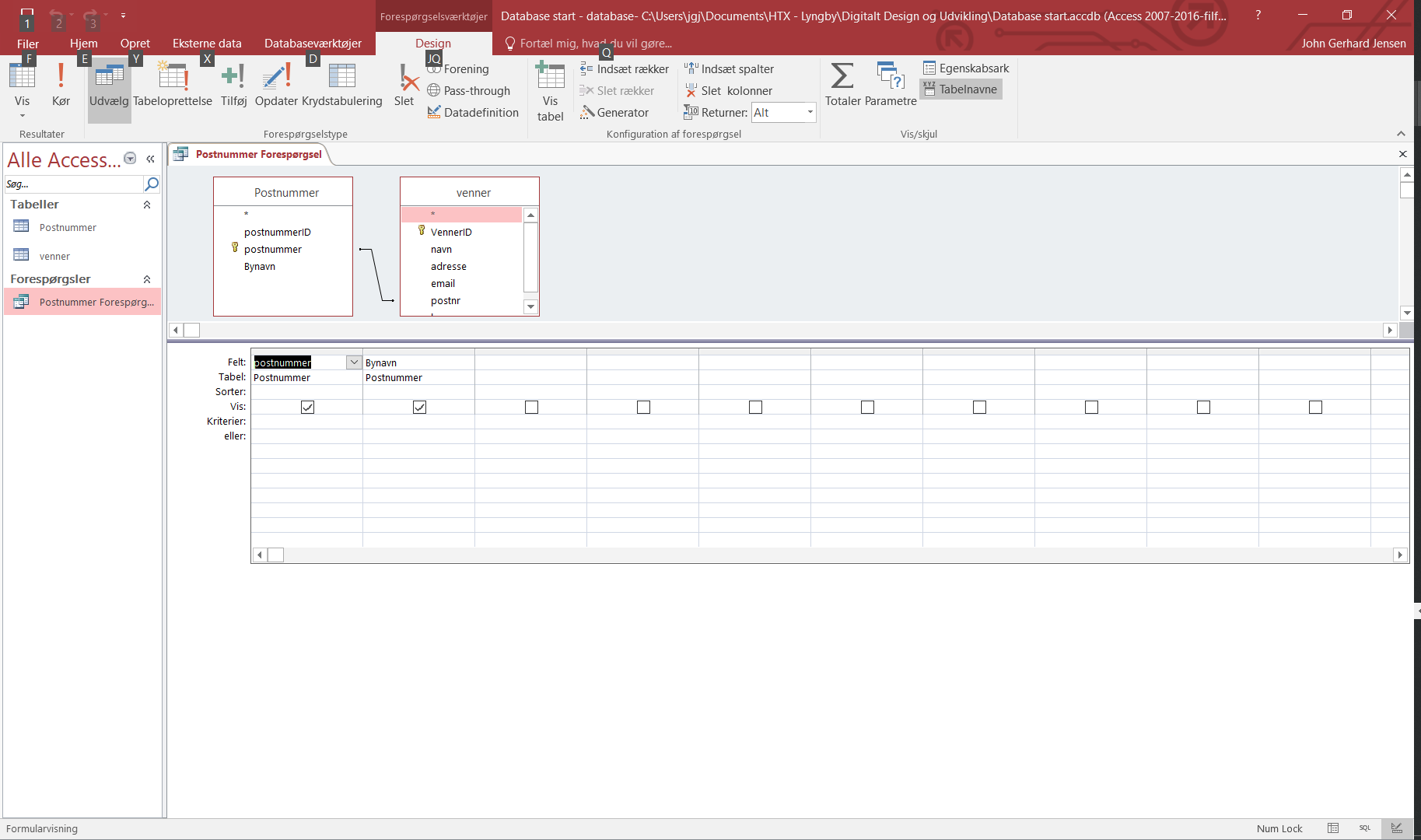
1:1 data er som regel data i tabellen, der er relateret til tabelnavnet. I min venner-tabel har jeg registreret navn, adresse, email og postnr om hver af mine venner.

1:m data er data som ikke er direkte relateret til tabelnavnet. Her er bynavn ikke afhængigt at min ven. Det hedder 2100 Kbh Ø, uanset om han bor der eller ej. Derfor skal koblingen 2100 – Kbh Ø skilles ud fra venner-tabellen og placeres i sin egen tabel.

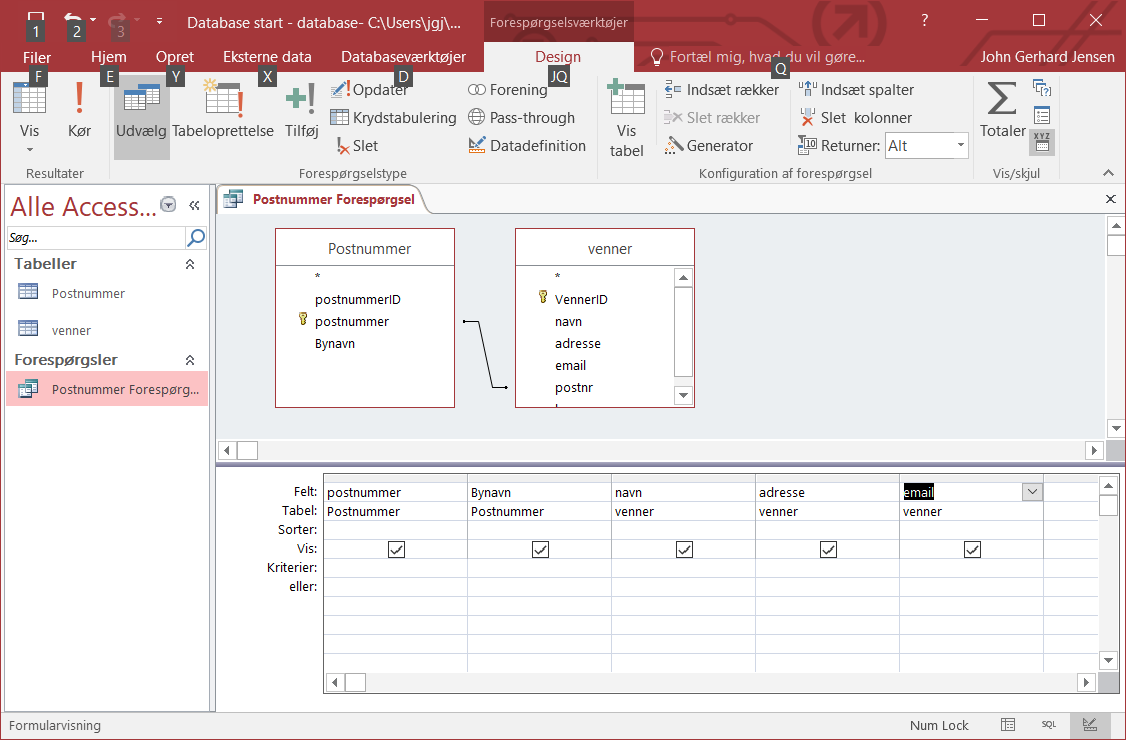
Vi starter med koble venner-tabellen sammen med postnrtabellen. Man laver en forespørgelse, hvor man tilføjer de to tabeller, se figur.



Der er redundans i ID-navnene, så de bliver koblet med en relation. Det er ikke godt. Det skal rettes til, så der ikke er nogen resundans i feltnavnene. relationen skal rettes til at være mellem postnummer i postnummertabellen og postnummer i vennertabellen



Figuren viser en forespørgelse, hvor vi har bedt om postnummer og bynavn, der skal mere data med, så vi beder om navn, adresse, postnummer og bynavn.



Nu ser resultatet således ud

