

Modellering og data



Snapchat

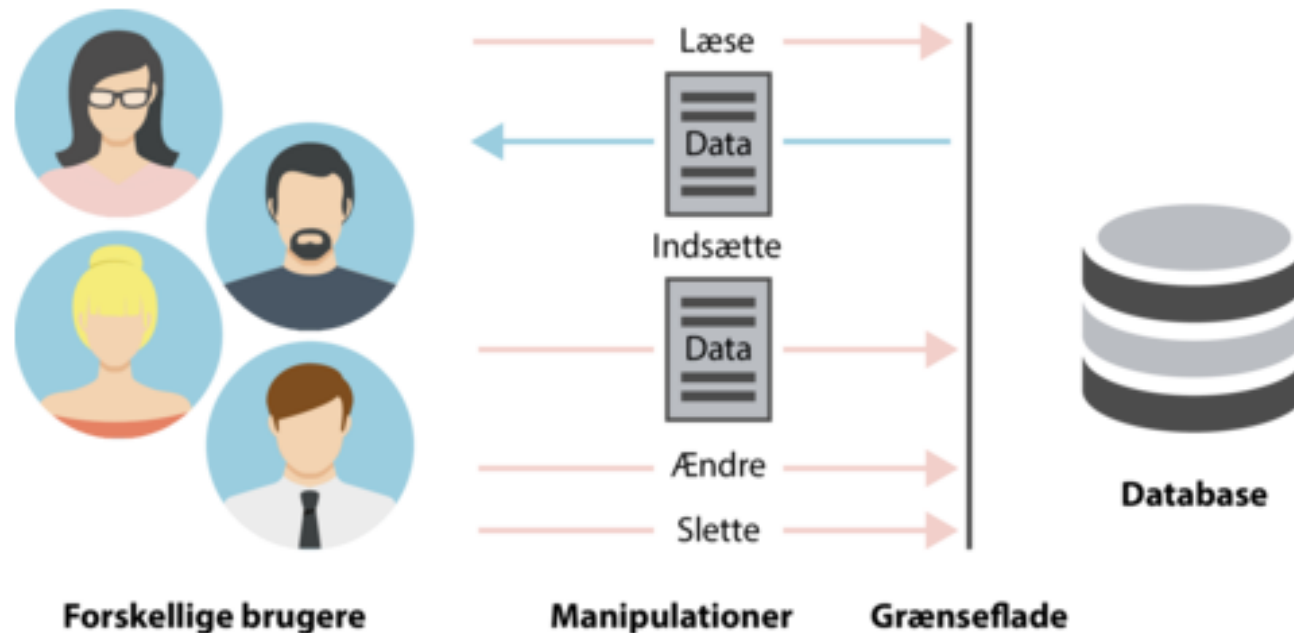
- Hvilke data har Snapchat brug for at gemme?
- Hvordan bliver data mon gemt/organiseret?



Hvad er en database?

- En **database** er en *struktureret* samling af data, som understøtter, at vi som brugere kan

- 1) Læse data
- 2) Indsætte data
- 3) Ændre data
- 4) Slette data



Kan vi ikke bare oprette en enkelt tabel i Excel til datalaget?

Kunder og konti hos Vejgaard Bank....

	A	B	C	D	E	F	G	H	I
1	Kundeld	Fornavn	Efternavn	Adresse	Postnummer	Kontold	Kontonavn	Kontotype	Saldo
2	1	Tove	Hansen	Hadsundvej 23, 1tv	9000	1	Nem	Nem	8756,31
3	1	Tove	Hansen	Hadsundvej 23, 1tv	9000	2	Budget	Budget	45891,98
4	1	Tove	Hansen	Hadsundvej 23, 1tv	9000	3	Opsparing Hus & Have	Opsparing	2345,72
5	1	Tove	Hansen	Hadsundvej 23, 1tv	9000	4	Forbrug	Forbrug	2540,59
6	2	Peter	Jensen	Petersborgvej 83	9000	5	Ting&Sager APS	Erhverv	89789,76
7	2	Peter	Jensen	Petersborgvej 83	9000	6	Nem	Nem	-234,87
8	2	Peter	Jensen	Petersborgvej 83	9000	7	Budget	Budget	34983,9
9	2	Peter	Jensen	Petersborgvej 83	9000	8	Forbrug	Forbrug	5634,98
10	3	Birgit	Sørensen	Hjørringvej 53, 2th	9400	9	Nem	Nem	3489,32
11	3	Birgit	Sørensen	Hjørringvej 53, 2th	9400	10	Opsparing Asger	Børneopsparing	23901,76
12	3	Birgit	Sørensen	Hjørringvej 53, 2th	9400	11	Forbrug Daglig	Forbrug	4535,01
13	3	Birgit	Sørensen	Hjørringvej 53, 2th	9300	12	Budget	Budget	23416,76
14	3	Birgit	Sørensen	Hjørringvej 53, 2th	9400	13	Bilkonto	Forbrug	-1898,7

NEJ! Mange gentagelser.

Er der nogen oplagte problemer ved datalaget hos Vejgaard Bank? Overvej!

- 1) Er det effektivt mht. plads?**
- 2) Hvordan oprettes en ny kunde?**
- 3) Kan du finde en fejl? Hvorfor kunne denne fejl nemt opstå?**

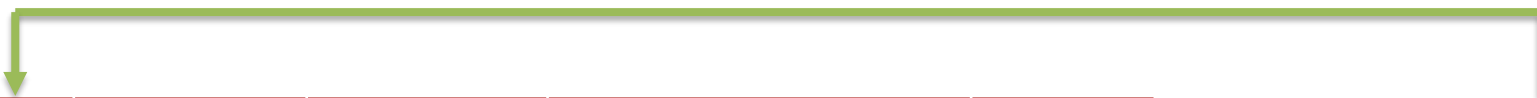
Problematisk, vi kan kun oprette en kunde, hvis der samtidig oprettes en konto, evt. en "tom konto"

Stor risiko for at begå fejl ved indsættelse af ny konto! Gad vide, hvad Birgits postnummer er?

Databasens struktur

Problemet opstod, da vi samlede to forskellige størrelser i én tabel, løsningen er:

- Der er to forskellige **størrelser/entiteter**: *kunde* og *konti*, dette giver to **tabeller**
- Hvert **datapunkt** svarer til en **række**
- **Kolonner** er **egenskaber/attributter** for *kunder* og *konti*
- Id er **unik** i hver række, dvs. i kunde – tabellen er der kun én række med Id = 1, én række med Id = 2, ...
- En konto bliver knyttet til en kunde via en **relation** (vi tilføjer blot en kolonne Kunde i kontotabellen)



Id	Fornavn	Efternavn	Adresse	PostNr
1	Tove	Hansen	Hadsundvej 23, 1tv	9000
2	Peter	Jensen	Petersborgvej 83	9000
3	Birgit	Sørensen	Hjørringvej 53, 2th	9400

Kundetabel

Id	Kunde	Navn	Type	Saldo
1	1	Nem	Nem	8756,31
7	2	Budget	Budget	34983,9
8	2	Forbrug	Forbrug	5634,98
12	3	Budget	Budget	23416,76
13	3	Bilkonto	Forbrug	-1898,7

Kontotabel

Databasesystemer

- Et ***databasesystem*** er et program, hvor vi kan oprette og arbejde med databaser (vi bruger *sqlite*).
 - Et godt databasesystem har bl.a. følgende kvaliteter
 - 1) Data lagres *pladseffektivt*
 - 2) Data lagres *sikkert*
 - 3) Data lagres *korrekt* (f.eks. to personer tilgår den samme konti, databasesystemet sørger da selv for at holde styr på at overførslerne gøres korrekt, og i den rigtige rækkefølge)
- Dette behøver vi ikke tænke over, når vi bruger et databasesystem!

SQL

- **S**tructured **Q**uery **L**anguage
- Det kommandosprog man bruger til at
 - oprette tabeller
 - indsætte rækker i tabeller
 - slette rækker fra tabeller
 - finde rækker som opfylder en bestemt betingelse
 - osv.
- Se <https://www.w3schools.com/sql/default.asp>

SQLiteBrowser

- Installer SQLiteBrowser på din computer via dette link:
<https://sqlitebrowser.org/dl/>
- Opret en ny mappe, som du kan finde igen!
- Hent filen VejgaardBank.db (ligger under "Præsentation") – og gem filen i den mappe, du lige har lavet. VENT!

SQL – læse fra tabeller

Vi afprøver følgende sammen i Execute SQL:

- Vælge alle kolonner og rækker fra en tabel:

```
SELECT * FROM Kunder;
```

- Vælge en eller flere kolonner fra en tabel:

```
SELECT fornavn, efternavn FROM Kunder;
```

- Filtrere data med *where* (udvælge data med betingelse)

```
SELECT fornavn, efternavn FROM Kunder  
WHERE Postnr = 9000;
```

**HUSK at
gemme SQL
filen og
databasefilen!**

SQL – læse fra tabeller

- Generelt er syntaksen

SELECT ← Hvilke kolonner vil du gerne se?

FROM ← Fra hvilken tabel?

WHERE ← Med hvilken betingelse?

SQL – læse fra tabeller – Øvelse

(herfra kører I selvstændigt)

Forklar nogle af
øvelserne (men ikke
dem alle!) i din logbog

Øvelse 1: Læs hele kontitabellen

Øvelse 2: Læs kolonnerne id og saldo fra kontitabellen

Øvelse 3: Find alle konti med en saldo på over 40000 (HINT: Betingelsen er saldo>40000)

Øvelse 4: Find alle konti, hvor saldoen er negativ.

SQL – læse fra tabeller - Øvelse

Øvelse 5: Forklar forskellen på

```
SELECT fornavn FROM Kunder;
```

```
SELECT DISTINCT fornavn FROM Kunder;
```

```
SELECT count(Postnr) FROM Kunder;
```

```
SELECT count(DISTINCT Postnr) FROM Kunder;
```

Øvelse 6: Udskriv de forskellige kontotyper

Øvelse 7: Tæl antallet af forskellige kontotyper (ved at bruge en af ovenstående kommandoer)

Ordne efter kolonne

Afprøv nedenstående i Execute SQL:

```
SELECT fornavn, efternavn FROM Kunder  
order by efternavn asc;
```

```
SELECT fornavn, efternavn FROM Kunder  
order by efternavn desc;
```

Øvelse 8: Udskriv id og saldo på alle konti, sorter i stigende og derefter aftagende orden efter saldo

SQL – indsætte

Generelt:

```
INSERT INTO tabel_navn (kolonne1, kolonne2, ..., ) VALUES (værdi1, værdi1, ...);
```

Eksempel 1

```
insert into Kunder(id, fornavn, efternavn, adresse, Postnr)
VALUES
(1, 'Tove', 'Hansen', 'Hadsundvej 22, 1.tv', 9000);
```

Eksempel 2

```
insert into Konti(id, kunde, navn, type, saldo)
VALUES
(1, 1, 'Nem', 'Nem', 8756.31),
(2, 1, 'Budget', 'Budget', 45891.98);
```

SQL – indsætte

Øvelse 9: Indsæt en ny kunde:

Søren Andersen
Hohøje 55,
9500 Hobro

Opret en nem konto og en forbrugskonto til Søren, hvor saldoerne som udgangspunkt er 0 kroner.

Øvelse 10: Prøv at køre kommandoen:

```
insert into Kunder(id, fornavn, efternavn, adresse, Postnr)
VALUES
(1, 'Børge', 'Henriksen', 'Hobrovej 57', 9000);
```

Forklar fejlmeldingen. Hvordan kan du få oprettet Børge som kunde?

SQL – slette data fra en tabel

Generelt:

```
DELETE FROM tabel_navn WHERE betingelse;
```

Eksempel. Vi sletter Birgits bilkonto ved

```
DELETE FROM Konti WHERE id=13;
```

Øvelse 11. Afprøv ovenstående. Du skal først læse hele tabellen. Kontroller derefter ved at læse hele tabellen.

Øvelse 12 Slet kunden Børge Henriksen, der blev oprettet på sidste slide.

Øvelse 13 Prøv at slette kunden Birgit Sørensen, hvad går galt? Kan du forklare?

SQL – opdatere tabel

Generelt

```
UPDATE table_name SET kolonne = value WHERE betingelse ;
```

Eksempler

```
UPDATE Kunder SET adresse = 'Øster Alle 45, 3 tv' WHERE id = 3;
```

```
UPDATE Konti SET saldo = saldo+1000 WHERE type = 'Nem';
```

Øvelse 14: Prøv at køre ovenstående kommandoer. Hvad er der sket? Du skal lige ændre saldoen, så den kommer tilbage til det den var før.

Øvelse 15: Søren Andersen indsætter 20000 på sin nem konto og overfører derefter 2000 til sin forbrugskonto. Foretag transaktionerne (brug id for hans konti).

SQL Wildcards

* betyder alle.

% betyder et vilkårligt antal karakterer.

% er ofte nyttigt ved strengsøgninger:

```
SELECT * FROM Kunder WHERE fornavn = 'Birgit';
```

```
SELECT * FROM Kunder WHERE fornavn is 'Birgit';
```

```
SELECT * FROM Kunder WHERE fornavn like 'B%';
```

```
SELECT * FROM Kunder WHERE fornavn like '%r%';
```

Øvelse 16: Afprøv ovenstående.

Øvelse 17: Find id, type og saldo på alle bankkonto af typen Forbrug, prøv forskellige måder. Udskriv sorteret efter saldo.

Booleske operatorer og mellem

Ved betingelserne bruges ofte AND, OR eller NOT (især AND 😊)

```
SELECT * FROM Kunder WHERE  
Postnr=9000 AND efternavn BETWEEN 'A' AND 'J';
```

```
SELECT * FROM Kunder WHERE  
Postnr=9000 OR Postnr=9400;
```

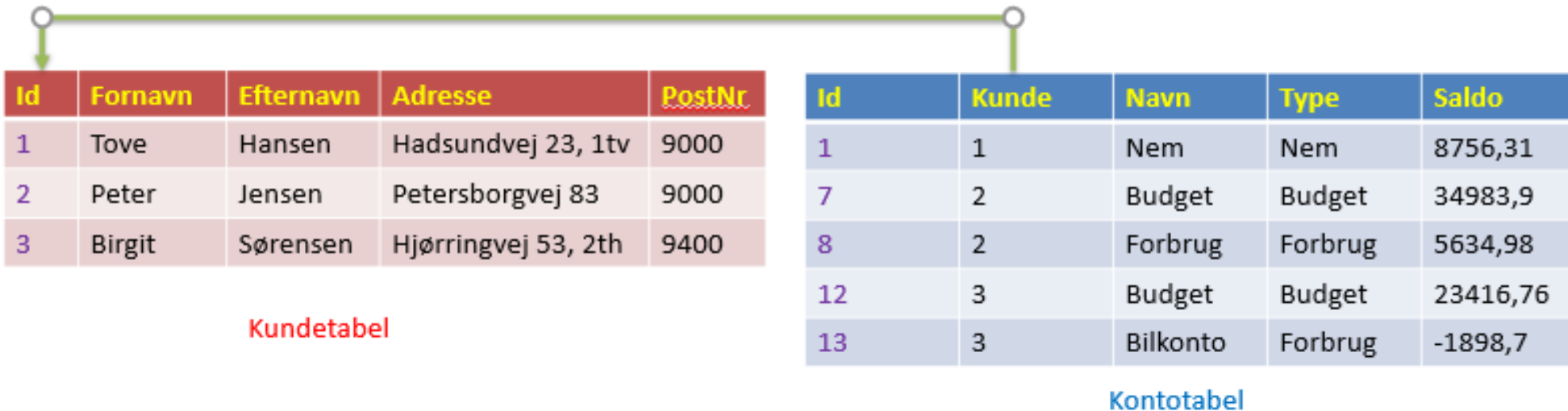
Øvelse 18: Afprøv ovenstående (se bank kommando).

Øvelse 19: Find alle bankkonti af typen Nem og med en saldo over 1000.

Øvelse 20: Find alle bankkonti der har en saldo mellem 10000 og 50000.

SQL – Koble tabeller

Indtil videre har vi jo ikke brugt de to tabeller på én gang, men dette er jo faktisk grunden til **relationen**



Opgaven er at finde Toves konti

Vi kan forsøge med at vælge alle data fra BEGGE tabeller ved

```
SELECT * FROM Kunder, Konti
```

SQL – Koble tabeller

Dette giver ALT for meget data. Alle rækker fra kunder parres med alle rækker fra konto. Altså i alt 3 gange 13 rækker, dvs. 39 rækker

id	fornavn	efternavn	adresse	Postnr	id	kunde	navn	type	saldo
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	1	1	Nem	Nem	8756.31
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	2	1	Budget	Budget	45891.98
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	3	1	Opsparing Hus & Have	Opsparing	2345.72
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	4	1	Forbrug	Forbrug	2540.5
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	5	2	Ting & Sager APS	Erhverv	89789.76
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	6	2	Nem	Nem	-234.87
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	7	2	Budget	Budget	34983.9
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	8	2	Forbrug	Forbrug	5634.98
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	9	3	Nem	Nem	3489.32
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	10	3	Opsparing Asger	Børneopsparing	23901.76
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	11	3	Forbrug Daglig	Forbrug	4535.01
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	12	3	Budget	Budget	23416.76
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	13	3	Bilkonto	Forbrug	-1898.7

SQL – Koble tabeller

Vi ser, at vi kan vælge betingelsen Kunde.id=Konti.kunde AND fornavn = 'Tove'

id	fornavn	efternavn	adresse	Postnr	id	kunde	navn	type	saldo
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	1	1	Nem	Nem	8756.31
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	2	1	Budget	Budget	45891.98
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	3	1	Opsparing Hus & Have	Opsparing	2345.72
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	4	1	Forbrug	Forbrug	2540.5
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	5	2	Ting & Sager APS	Erhverv	89789.76
1	Tove	Hansen	Hadsundvej 22, 1.tv	9000	6	2	Nem	Nem	-234.87

Dette giver følgende søgninger (vi prøver dem af)

```
SELECT * FROM Kunder,Konti
```

```
WHERE Kunder.id = Konti.kunde AND fornavn = 'Tove';
```

```
SELECT fornavn,navn,saldo FROM Kunder,Konti
```

```
WHERE Kunder.id = Konti.kunde AND fornavn = 'Tove';
```

Øvelse 21 Find fornavn, efternavn, kontonavn og saldo på alle kunder der bor i Aalborg.

Beregninger

Databasen indeholder jodata😊 Det er naturligt at foretage beregninger, her er lidt blandede eksempler

```
SELECT count(*) FROM Kunder;
```

```
SELECT id,navn,max(saldo) FROM konti;
```

```
SELECT avg(saldo) FROM konti;
```

```
SELECT sum(saldo) FROM konti;
```

Øvelse 22: Afprøv ovenstående (se bank kommando).

Øvelse 23: På hvilken af Birgits konti står det største beløb?

Øvelse 24: Hvor meget har Tove i gennemsnit på alle sine konti?

Øvelse 25: Hvor meget har Peter i alt stående?

Øvelse 26: Hvem er ejeren af saldoen med det største beløb?

Øvelse 27: Hvilken saldo har det mindste beløb stående?