# Module 1: Til- og fraflyttere

DAT18C - Gruppe 3

Frederik Lundbeck Jørgensen
Tobias Midskard Sørensen
Tariq Zamani
Anders Ahrenkilde

**Hand in. 1.3.2019 kl. 16:00**

**GitHub**: https://github.com/Zamanien/ModulOpgave_1

COPENHAGEN SCHOOL OF DESIGN AND TECHNOLOGY

## Indholdsfortegnelse:

# Problemstilling

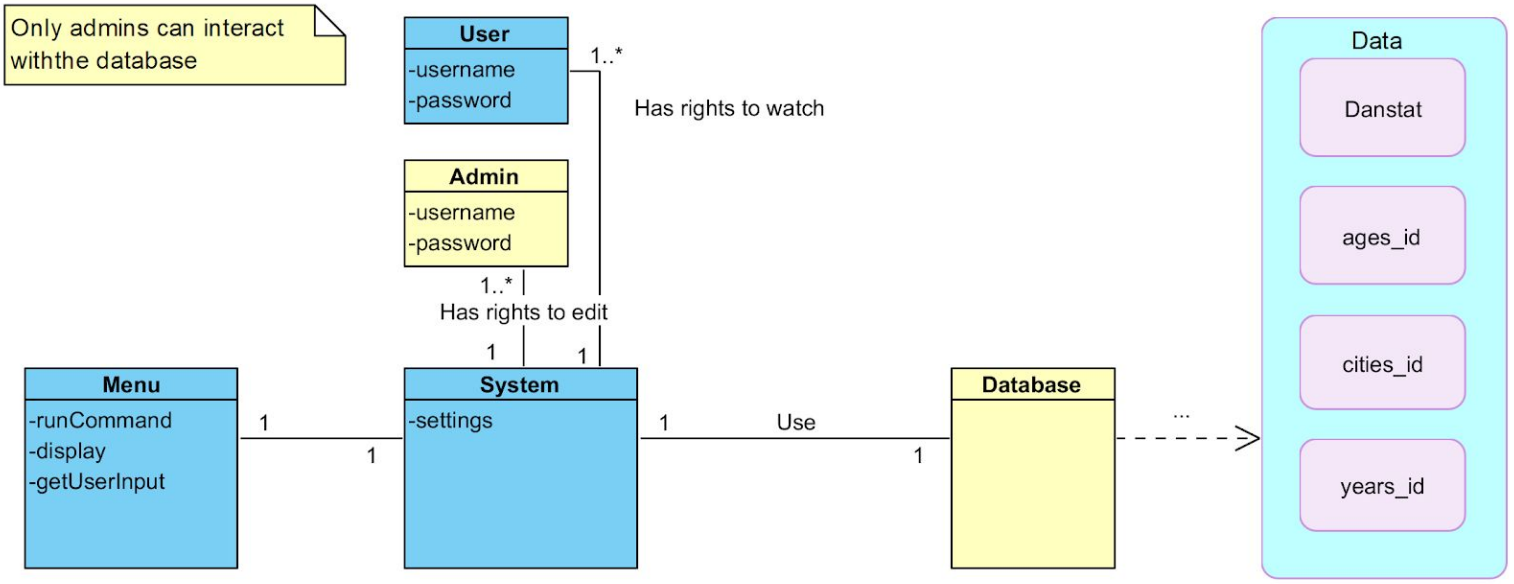
Rå data om beboelse og til- og fraflytning skal organiseres i en database. Datasættet skal være organiseret på den mest effektive måde. Samtidig skal dataen også smides op på en database.

# Afgrænsning

Vi har valgt kun at fokusere på de rå data om til- og fraflytning. Hertil vil vi udvikle et brugergrænseflade, som køres i terminalen. Brugeren skal have adgang til at søge efter informationer omkring til- og fraflyttere mellem kommunerne i Danmark.

# Domæne model

Domænemodellen giver et overordnet udtryk af databasen. Her kan det ses, at kun admin har tilladelse til, at redigere i databasen, hvorimod user kun har rettigheder til at søge i Datasættene.



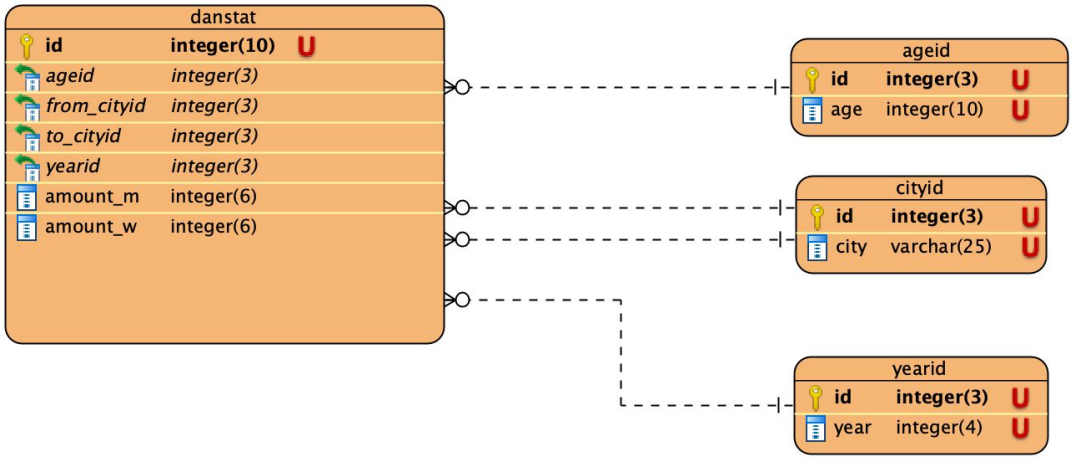
# ER-diagram

Vores databasedesign bygger sig op omkring én hovedtabel. Denne tabel er understøttet af 3 indeksering tabeller, som hver kun har en hovednøgle, og dertil en værdi på de forskellige kategorier med hhv. aler, by og årstal.

Vi har lavet en transponering af de originale årstal, som var repræsenteret som kolonner. På den måde skal vi ikke oprette en ny kolonne for hvert nyt år.  Hertil blev vores antal af rækker 11 gange så mange. For at mindske antallet af rækker opretter vi to nye kolonner, én hver for mænd og kvinder, da de begge er afhængige af en identisk kombination.

Den store tabel *danstat,* er dybest set en sammensmeltning af de to originale excel-filer. De rå data er bundet sammen, ved at der er tre variabler, som tilsammen kendetegner unik data. Heraf har vi *alder, fraflytnings- og tilflytningskommune og sidst et årstal*, som definerer dataene for flyttende mænd og kvinder.

Se vores ER-diagram herunder.



# 3NF

Efter at vi havde realiseret vores tanker. Fulgte vi 3NF-teknikken:

1. Vi havde ingen redundant data.
2. Vores *amount_m* og *amount_w* er vores datafelter. Vi kan ikke udskille nogle af nøglerne, og derfor er vores datafelter afhængige af alle sammensatte nøgler.
3. Der findes ingen felter uden for primærnøglen, som er indbyrdes afhængige.

Vores ERD opfylder derfor 3NF.

# Importering af data

En stor del af vores tid er blevet brugt på at få dataen ind i databasen. Vi brugte 4 hele moduler på at snakke alle tingene igennem. Tankerne gik fra hvordan at tabellerne skulle koordineres, til hvordan at vi skulle importere. Det var vigtigt for os at tabellerne var få effektive som muligt, og dermed havde vi mange bekymringer omkring perfomance ifb. med hvordan at MySQL håndtere data.

Vores fremgangsmåde for importering af data:

1. Det ene arks rå data åbnes i excel.
2. Årstal vendes vha. pivottabeller.
3. Det andet ark åbnes og årstal vendes.
4. til- og fraflytningskollonner byttes rundt, så de matcher det andet ark.
5. Sammensættelse af de to ark.
6. Mænd og kvinder får en kollonne hver.
7. Uploades til database serveren vha. *import wizard.*

Hertil kan vi nu benytte SQL kommandoer til at sprede dataen ud i de fremstillede tabeller (*danstat, ageid osv.*). Først skulle vi oprette de forskellige indekseringstabeller. Herefter kunne vi bruge dem, til at sætte de forskellige kategorier ind, repræsenteret som foreign keys.

```
INSERT INTO ageid
SELECT DISTINCT ages FROM imported_raw_data;

INSERT INTO yearid
SELECT DISTINCT years FROM imported_raw_data;

INSERT INTO cityid
SELECT DISTINCT city FROM imported_raw_data;

INSERT INTO danstat (amount_w, amount_m, year_id, age_id, to_cityid,
from_cityid)
SELECT men, women, yearid.id, ageid.id, cityid.id AS tilKommune, cityid.id AS
fraKommune FROM imported_raw_data
INNER JOIN yearid ON aar = Aar.aarstal
INNER JOIN ageid ON alder = Alder.aldersgruppe
INNER JOIN cityid ON tilflytningskommune = Kommune.tilKommune_navn
INNER JOIN cityid AS fraKommune ON imported_raw_data =
fraKommune.kommune_navn
```

# Installation af java-program

Hvis man bare vil køre projektet med den comilede executable, køres denne kommando.

```
$ git clone https://github.com/Zamanien/ModulOpgave_1

$ java -jar DatabaseApp_Final.jar
```

Der kræves Maven til at compile vores projekt. (https://maven.apache.org/install.html)

```
$ git clone https://github.com/Zamanien/ModulOpgave_1

$ cd ModulOpgave_1/DatabaseApplication

$ mvn clean compile assembly:single

$ java -jar target/DatabaseApplication-1.0-jar-with-dependencies.jar
```

# Konklusion

I opgaven er der blevet etableret en database hvori oplysninger om til- og fraflyttet i samtlige Københavns kommuner indgår. Der er primært fokuseret på datasættene "*Danstat_FLY66_FRA"* samt "*Danstat_FLY66_TIL".* Dette valg er foretaget på baggrund af tidsmangel.

De to datasæt er blevet omstruktureret med formålet, at formindske mængden af kolonner og dermed øge muligheden for, at tilføje, redigere eller slette rækkerne. Som resultat vil databasen være mere driftsikker og dynamisk.

Dertil er der blevet lavet en brugergrænseflade, som kører gennem CMD. Heri kan brugeren så - afhængig af om vedkommende er admin eller guest - få adgang til databasen og kigge/redigere i databasen (CRUD).