Normalisering

Øvelse 1: Terminologi

Forventning: At forstå grundlæggende begreber inden for normalisering, herunder anomalier, funktionelle afhængigheder, referentiel integritetsrestriktion, normalformer og normaliseringsproces.

*Tidsramme: 3**5** minutter*
Sørg for, at alle får mulighed for at tale.

Brug **Ordet rundt** til at reflektere over begreberne:

- **Beskriv de tre typer af dataanomalier og giv eksempler på hver type fra de projekter, du har arbejdet med.**

  Dataanomalier opstår, når en database ikke er ordentligt normaliseret. Der er tre hovedtyper af dataanomalier:
  - **Indsættelsesanomalier**: Dette opstår, når det er umuligt at indsætte data uden samtidig at indsætte unødvendige eller ufuldstændige data. Eksempel: I et projekt, hvor en database med kunder og ordrer ikke er normaliseret, kan det være, at man ikke kan indsætte en ny kunde uden også at oprette en ordre.
  - **Opdateringsanomalier**: Hvis de samme data er duplikeret flere steder, kan en ændring i ét sted kræve, at man opdaterer data flere steder, hvilket kan føre til inkonsistens, hvis man glemmer det. Eksempel: Hvis en kundes adresse opbevares i flere tabeller, og den ændres ét sted men ikke i de andre, opstår inkonsistens.
  - **Sletningsanomalier**: Sletning af data kan utilsigtet føre til tab af andre vigtige data. Eksempel: Hvis et projekt bruger en tabel, der indeholder både oplysninger om produkter og leverandører, kan sletning af et produkt samtidig slette en leverandørs oplysninger, selvom leverandøren stadig eksisterer.

- **Forklar begrebet funktionel afhængighed med egne ord og giv eksempler på funktionelle afhængigheder.**

  Funktionel afhængighed refererer til relationen mellem attributter i en tabel, hvor værdien af én attribut (eller en gruppe af attributter) bestemmer værdien af en anden attribut.
  Eksempel: I en kundedatabase kan feltet *kundenummer* bestemme *kundenavn*. Det betyder, at *kundenavn* er funktionelt afhængig af *kundenummer*.

- **Forklar begrebet referentiel integritetsrestriktion med egne ord og giv eksempler på den.**

  Referentiel integritet handler om at sikre, at relationer mellem tabeller bevares konsistente. Det indebærer, at hvis én tabel henviser til en række i en anden tabel (f.eks. via en fremmednøgle), så skal den henviste række faktisk eksistere.

- **Forklar begrebet normalisering, samt dens fordele i database design og beskriv de tre første normalformer med eksempler.**

Normalisering er en proces i database design, der bruges til at organisere data på en måde, der reducerer redundans og forbedrer dataintegritet. Normalisering indebærer at opdele en stor tabel i mindre, relaterede tabeller og sikre, at data kun opbevares én gang.

- **Første normalform (1NF)**: En tabel er i 1NF, hvis alle dens attributter indeholder atomare (uopdelte) værdier. Eksempel: En tabel med ordrer, hvor der kun er ét produkt per række, og hvor man ikke har opdelte eller gentagne grupper i én celle.
- **Anden normalform (2NF)**: En tabel er i 2NF, hvis den er i 1NF og alle ikke-nøgle attributter er fuldstændig afhængige af hele den primære nøgle, ikke kun en del af den. Eksempel: Hvis vi har en tabel med ordrer, hvor primærnøglen er (ordrenummer, produkt-ID), og vi har en kolonne med produktbeskrivelse, som kun er afhængig af produkt-ID'et, bør denne kolonne flyttes til en separat tabel.
- **Tredje normalform (3NF)**: En tabel er i 3NF, hvis den er i 2NF, og der ikke er nogen transitive afhængigheder (attributter, der er afhængige af andre ikke-nøgle attributter). Eksempel: Hvis vi har en tabel med medarbejdere, hvor både medarbejderens afdeling og afdelingens leder gemmes, bør oplysninger om afdelingslederen flyttes til en separat tabel, da de er afhængige af afdelingen, ikke af medarbejderen.

- **Forklar trinene i normaliseringsprocessen og tegn en model, der viser sammenhæng mellem normalformer, anomalier og funktionelle afhængigheder. Modellen uploades på Ahaslides gruppevis.** Grupperne rater de andres modeller og den bedste model fremvises i vidensdeling kl. 14:30.

Identifikation af attributter: Saml alle de nødvendige datafelter og identificer funktionelle afhængigheder.

Opdeling af tabeller: Bryd tabeller op for at sikre, at hver tabel har attributter, der er direkte relateret til primærnøglen.

transitive afhængigheder: Flyt afhængige attributter til nye tabeller for at opnå højere normalformer.

Øvelse 2: Fra Excel til UML-database model (Pet Paradise)

Forventning: At konvertere et regneark med data fra dyreklinikken Pet Paradise til en UML-databasemodel gennem en domænemodel og normaliseringsprocessen.

*Tidsramme:7**5** minutter*

Dyreklinikken **Pet Paradise** har mere eller mindre desperat henvendt sig til dig for at få hjælp til deres mest centrale arbejdsgang. Klinikken anvender et regneark som omdrejningspunkt for registrering af alle deres kunder (både dyr og ejere) samt evt. udførte behandlinger. Efter et par 'uheld' med regnearket, er klinikken godt træt af kun at have dette ene regneark, som alle medarbejdere skal opdatere (mange gange samtidigt), når der skal registreres kunder og behandlinger. Du skal lave en mere holdbar løsning til dem, så klinikken ikke ender med at være Pet Hell i stedet for.

Ud fra det udleverede regneark skal du udvikle en fleksibel databaseløsning ved at udarbejde en domænemodel og derefter designe en UML-databasemodel gennem normaliseringsprocessen.

Benyt den CL-struktur, du finder bedst egnet til de følgende øvelser.

Øvelse 2.1: Problemer med regnearket

Pet Paradise kan ikke altid forstå hvilke problemer, regnearket giver, kun at det ofte går galt, når flere medarbejdere skal bruge det. Du skal nu identificere potentielle problemer med at vedligeholde regnearket, så du kan forklare det til klinikken og foreslå en bedre løsning.

Bemærk, at enheden på 'PetWeight'-kolonnen er pund, og enheden på Charge-kolonnen er US dollars.
Udfør følgende:

- Analyser regnearket for at forstå de forskellige attributter, data og deres relationer.
- Analysér regnearket og find de potentielle udfordringer.
  De samme Owner-detaljer (OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail) bliver gentaget for hver Pet.
  Det er redundans og kan lede til datainkonsistens.

- Identificér indsættelses-, opdaterings- og sletteanomalier i regnearket.
  Indsættelesanormalier
  Hvis en Pet bliver tilføjet uden en Service fra starten, kan der være problemer med at indsætte da Service, Date, and Charge er obligatoriske felter i denne struktur

  Opdateringsanormalier
  Hvis en Owner's kontaktinformationer ændres, skal det opdateres flere steder

  Sletteanormalier
  Hvis man sletter den sidste Service for en Pet, mister man vigtig information omkring det Pet eller dets Owner (Ingen seperate tabeller for Pets og Owners).

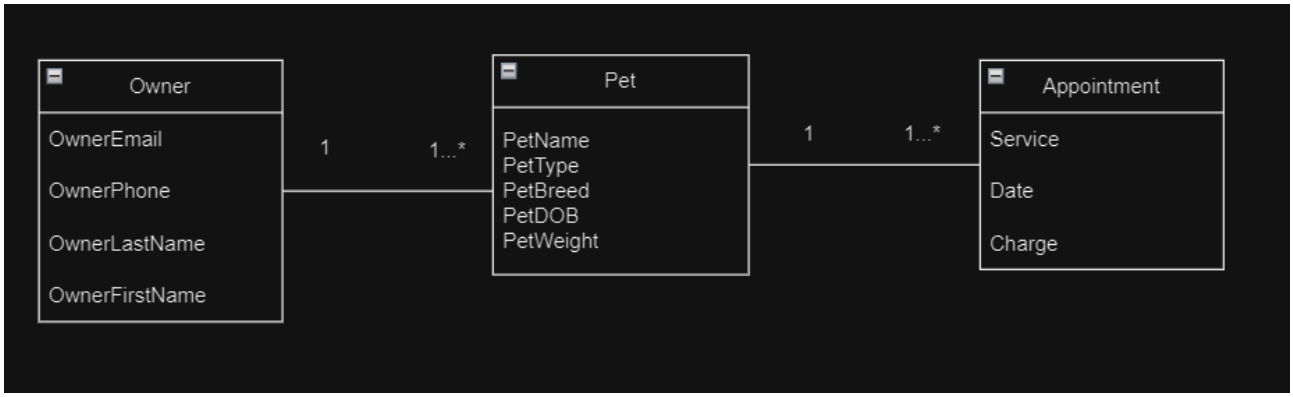
- Identificér alle funktionelle afhængigheder i regnearket.
  (OwnerPhone) → OwnerLastName, OwnerFirstName, OwnerEmail
  (PetName, OwnerPhone) → PetType, PetBreed, PetDOB, PetWeight
  (PetName, OwnerPhone, Service, Date) → Charge

Øvelse 2.2: Fra regneark til domænemodel

Gennemse regnearket og dan dernæst en domænemodel ud fra dets data. Udfør følgende:

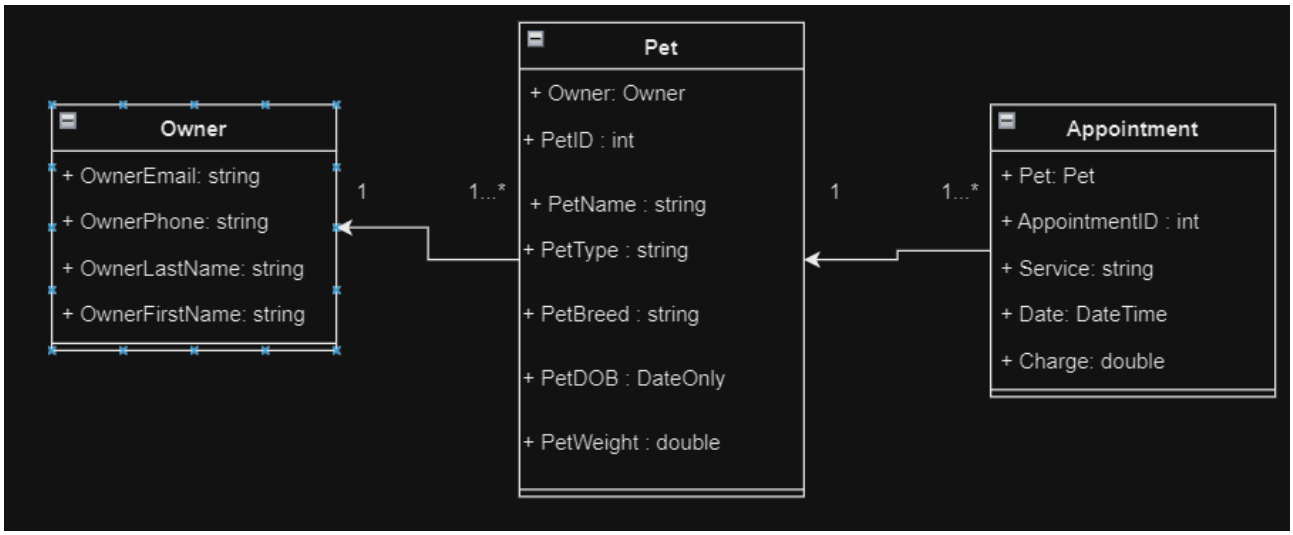
- Overvej hvilke kolonner i regnearket hører sammen, dvs. har samme tema, og dan en domæneklasse for hver af disse temaer. Fastlæg også relationerne inkl. kardinalitet (multiplicitet) mellem domæneklasserne
- Tegn domænemodellen i et passende grafisk værktøj (UMLet eller lign.)

https://app.diagrams.net/?src=about#Wb!2ZvumUSy5kaTxX9DdntarkwBS9uu9EFMqt0PTZeH00WpAHjXfAitRIPCBFyVKFFi%2F01U5PXUQIMSNVYOZYHXVCI6H3LADNWLDQZ#%7B%22pageId%22%3A%22-i0K5EUyWVyTgeTEeOy8%22%7D



Øvelse 2.3: Fra domænemodel til DCD

Du skal nu specificere en DCD for din domænemodel, hvor du specielt overvejer hvilke datatyper de enkelte attributter skal have og evt. navngivning.
Tegn DCD'et i et passende grafisk værktøj.



Øvelse 2.4: Fra DCD til relationelt databaseskema

Du skal nu udvikle et relationelt databaseskema ud fra domænemodellen, hvor du anvender **tekstnotationen** til at angive tabellerne samt deres felter, primære nøgler og fremmednøgler.

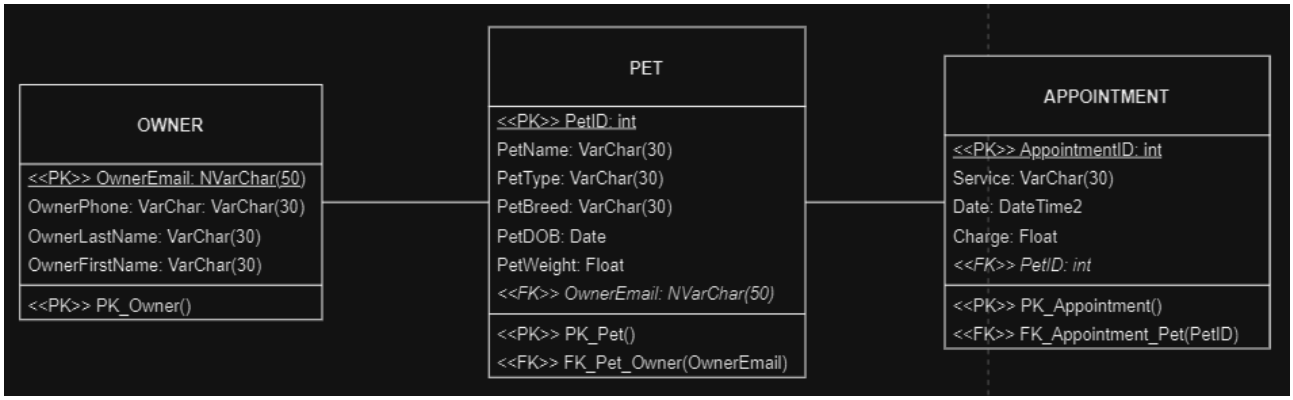
Benyt fremgangsmåden angivet i Database Design læringsobjektet og udfør følgende:

- Identificér alle tabeller nødvendige til at repræsentere din DCD fra forrige øvelse
- For hver tabel:
  - beskriv alle dets attributter
  - Angiv primærnøgler og fremmednøgler til at repræsentere alle relationer mellem tabeller

    OWNER(OwnerEmail OwnerPhone, OwnerLastName, OwnerFirstName)
    PET(PetID, PetName, PetType, PetBreed, PetDOB, PetWeight, *OwnerEmail*)
    SERVICERECORD(AppointmentID, Service, Date, Charge, *PetID*)

Øvelse 2.5: Fra relationelt databaseskema til UML-database model

Næste trin er at udarbejde et relationelt databaseskema, som grafisk illustrerer alle tabeller, felter, nøgler samt relationer mellem tabeller.

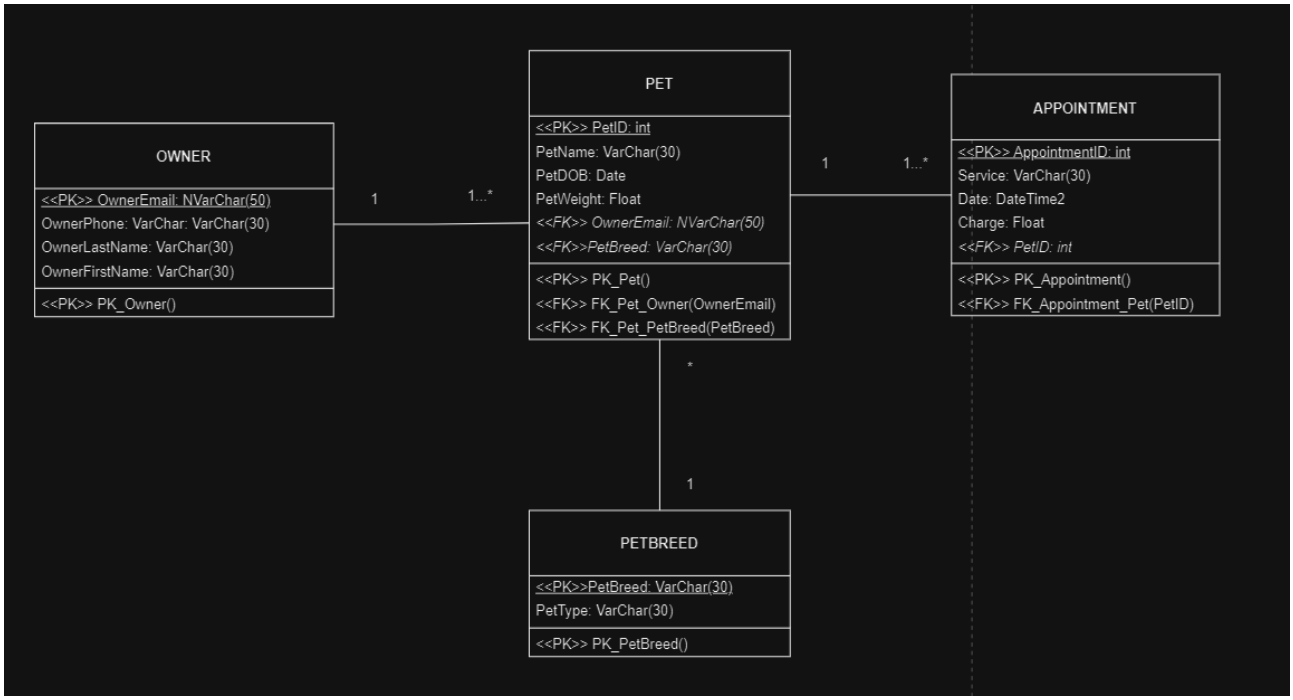

OWNER
<<PK>> OwnerEmail: NVarChar(50)
OwnerPhone: VarChar: VarChar(30)
OwnerLastName: VarChar(30)
OwnerFirstName: VarChar(30)
<<PK>> PK_Owner()
PET
<<PK>> PetID: int
PetName: VarChar(30)
PetType: VarChar(30)
PetBreed: VarChar(30)
PetDOB: Date
PetWeight: Float
<<FK>> OwnerEmail: NVarChar(50)
<<PK>> PK_Pet()
<<FK>> FK_Pet_Owner(OwnerEmail)
APPOINTMENT
<<PK>> AppointmentID: int
Service: VarChar(30)
Date: DateTime2
Charge: Float
<<FK>> PetID: int
<<PK>> PK_Appointment()
<<FK>> FK_Appointment_Pet(PetID)

Øvelse 2.6: Fra UML-database model til normaliseret database model

Sidste trin er at normalisere UML-databasemodellen til at kvalitetssikre din design.

Benyt fremgangsmåden angivet i normaliseringslæringsobjektet og udfør følgende:

- Sikr at tabellerne opfylder kravene til 1NF. Find tabeller, der ikke har atomare værdier og får dem i 1NF.
- Find afhængigheder i tabellerne og fjern delvise afhængigheder, så alle ikke-nøgleattributter er fuldt afhængige af hele primærnøglen til at alle tabeller opfylder kravene til 2NF.
- Find afhængigheder i tabellerne efter de er i 2NF og fjern transitive afhængigheder, så ikke-nøgleattributter kun afhænger af primærnøglen til at alle tabeller opfylder kravene til 3NF.
- Tegn en UML-databasemodel, der viser de normaliserede tabeller og deres relationer i et passende grafisk værktøj.



Øvelse 3: Normalisering af Webshop database

Forventning: At sikre at alle relationsskemaer i din webshop database er i tredje normalform (3NF) ved at gennemgå og anvende normaliseringsprocessen.

*Tidsramme:4**5** minutter*

Du og dit team har lavet en UML-database model til webshopcasen i sidste uge. Din opgave er nu at gennemgå hver tabel og sikre, at den opfylder kravene til 3NF. Dette indebærer at identificere og fjerne eventuelle anomali-situationer samt at håndtere funktionelle afhængigheder korrekt.

Benyt fremgangsmåden angivet i Normaliserings læringsobjektet og udfør følgende:
(For hver normaliseringsproces, du udfører, dokumentér ændringerne. Forklar, hvorfor hver ændring er nødvendig og hvordan det forbedrer databasens design)

- Start med at gennemgå hvert relationsskema **individuelt**. Noter alle attributter og identificer de eksisterende funktionelle afhængigheder.
- **Som team**, diskuter de funktionelle afhængigheder hver især har fandt på og anvend normaliseringsprocessen for at sikre, at de er i 3NF, dvs. opdel skemaerne for at fjerne de uønskede afhængigheder for at få databasen i en normaliseret tilstand.

Øvelse 4: Design af Viewlaget i C# (Webshop case)

Forventning: At designe og implementer brugergrænsefladerne i View laget for jeres webshop-applikation, du og dit team er startet at implementere i sidste uge. Denne opgave er betegnet som sprint 2 og indeholder planlægning og design af viewlaget.

*Tidsramme: **60** minutter*

Trin-for-trin Fremgangsmåde

Planlægning:

- Diskuter de vigtigste funktioner, som webshoppen skal tilbyde.
- Lav en liste over de nødvendige brugergrænseflader (UI-skærme) såsom login, produktvisning, indkøbskurv, checkout osv.

Design:

- Design de enkelte brugergrænseflader ved at lave skitser eller mockups. Dette kan gøres ved hjælp af værktøjer som Figma, Sketch, eller papirskitser. Hvert teammedlem skal designe brugergrænseflader for de klasser, som du har implementeret i modellaget i sidste uge.
- Definer, hvordan data vil blive præsenteret i hver UI og hvordan brugeren vil interagere med dem.

Implementering:

- Implementer designet i C# ved brug af WPF. Opret de nødvendige XAML-filer for WPF.
  (Det er ikke forventet nu at lave databinding, men den er bonus opgave til de hurtige).

Kvalitetssikring:

- Test de implementerede UIs for at sikre, at de opfylder kravene I har diskuteret i planlægningen.
- Gennemfør en gruppediskussion for at evaluere designet og få feedback fra teammedlemmer.

Øvelse 5: Inspektion af en konceptuel klasse (Bonus)

Forventning: At identificere potentielle problemer i en konceptuel klasse fra en domænemodel.

*Tidsramme: **15** minutter*

Hos virksomheden SillyCompany vil man gerne udarbejde et relationsskema, så de har udvalgt nedenstående konceptuelle klasse fra deres domænemodel. Lav en visuel inspektion af artefaktet og se, om du og dit team kan spotte nogle problemer ved at overveje hvordan ændringer i attributter kan påvirke dataintegriteten og konsistensen.

Spørgsmål til at motivere den rette dialog i teamet:

- Er det et problem, hvis biografnavnet ændrer sig?
  Det kan medføre inkonsistens i systemet, især hvis det optræder mange gange i forskellige bookinger. Hvis biografnavnet ændres, skal alle relaterede poster opdateres, hvilket kan være fejlagtigt og besværligt.

  Løsning
  Adskild biografoplysninger i en separat klasse, fx Biograf. Biograf kan have et unikt biografID, og BiografFilmForestillingBooking kan referere til Biograf via dette ID. Dette sikrer, at ændringer i biografens navn kun skal foretages ét sted.

- Er det et problem, hvis filmtitlen ændrer sig?
  Det samme problem og løsning som ovenover

- Er det et problem, hvis man ønsker at omdøbe en filmgenre?
  Ditto.

- Er der et problem, hvis man vil have filmen på flere forskellige tidspunkter?
  Hvis en film skal vises på flere tidspunkter, skal der være en måde at håndtere disse forskellige tidspunkter.
  I den nuværende klasse kan det være svært at adskille forskellige forestillinger.

  Løsning
  Seperat Forestilling-entitet. Forestilling kan så referere til Film via et FilmID. (1-M)

- Er der andre problemer med denne klasse?

Redundans: Hvis du har flere bookinger for samme film, biograf og forestilling, kan du ende med redundant data.

Skalerbarhed: Klassen kombinerer forskellige koncepter (biograf, film, forestilling, booking) i én klasse, hvilket kan føre til en uoverskuelig struktur og vanskeligheder med at udvide eller ændre systemet.

Mangel på normalisering: Klassen er ikke normaliseret. I et godt design bør du adskille klasser i entiteter, der hver især har en enkelt ansvarlighed og klare relationer.

Normalisering angivet som Databaseskema (hvad jeg lige kan overskue).

Med 1-M kardinalitet mellem MOVIE og GENRE

CINEMA(CinemaID, Name, City)

MOVIE(MovieID, Title, Duration, Director, PublicationDate, *GenreID*)

Genre(GenreID, Name)

SHOW(ShowID, PremierDate, TimeOfDay, MovieID, *CinemaID*)

BOOKING(BookingID, Email, PhoneNumber, *ShowID*)

Med M-M kardinalitet mellem MOVIE og GENRE

CINEMA(CinemaID, Name, City)

MOVIE(MovieID, Title, Duration, Director, PublicationDate)

Genre(GenreID, Name)

MOVIE_GENRE(*MovieID*, *GenreID*)

SHOW(ShowID, PremierDate, TimeOfDay, MovieID, *CinemaID*)

BOOKING(BookingID, Email, PhoneNumber, *ShowID*)

Øvelse 6: Vidensdeling kl. 14:30

Forbered og fremvis en præsentation af jeres bedste model, der fik mest rating i Øvelse 1 for klassen. Derudover fremvis og forklar de normaliseret databasemodel for Pet Paradise og webshop cases.

*Tidsramme: **30** minutter*