Handin 2_2

I4DAB

Hold 18

Navn:	Studie nr:
Kristoffer Stampe Villadsen	201607406
Malthe Dalgaard Jensen	201607413

Video: ” /Videoer/F18I4DABH22Video”

Underviser:
Jesper Tørresø

Contents

1 Domænemodel - (DDD)

I forbindelse med analysen af kravene stillet til opgaven blev der udarbejdet en domænemodel til identificering af domænerelevante klasser samt relationerne mellem disse. På figur 1 ses domænemodellen. Følgende klasser og deres relationer er blevet opstillet:

- **Person**
  - Kan have 0-1 **Email adresse**
  - Har 1 til mange **Telefonnummer**
  - Kan have 0 til mange **Adresse**
- **Adresse**
  - Tilhører 1 **By/postnummer**
  - Har 0 til mange **Person**
- **By/postnummer**
  - har 1 til mange **Adresse**
- **Email adresse**
  - Tilhører 1 **Person**
- **Telefonnummer**
  - Tilhører 1 **Person**

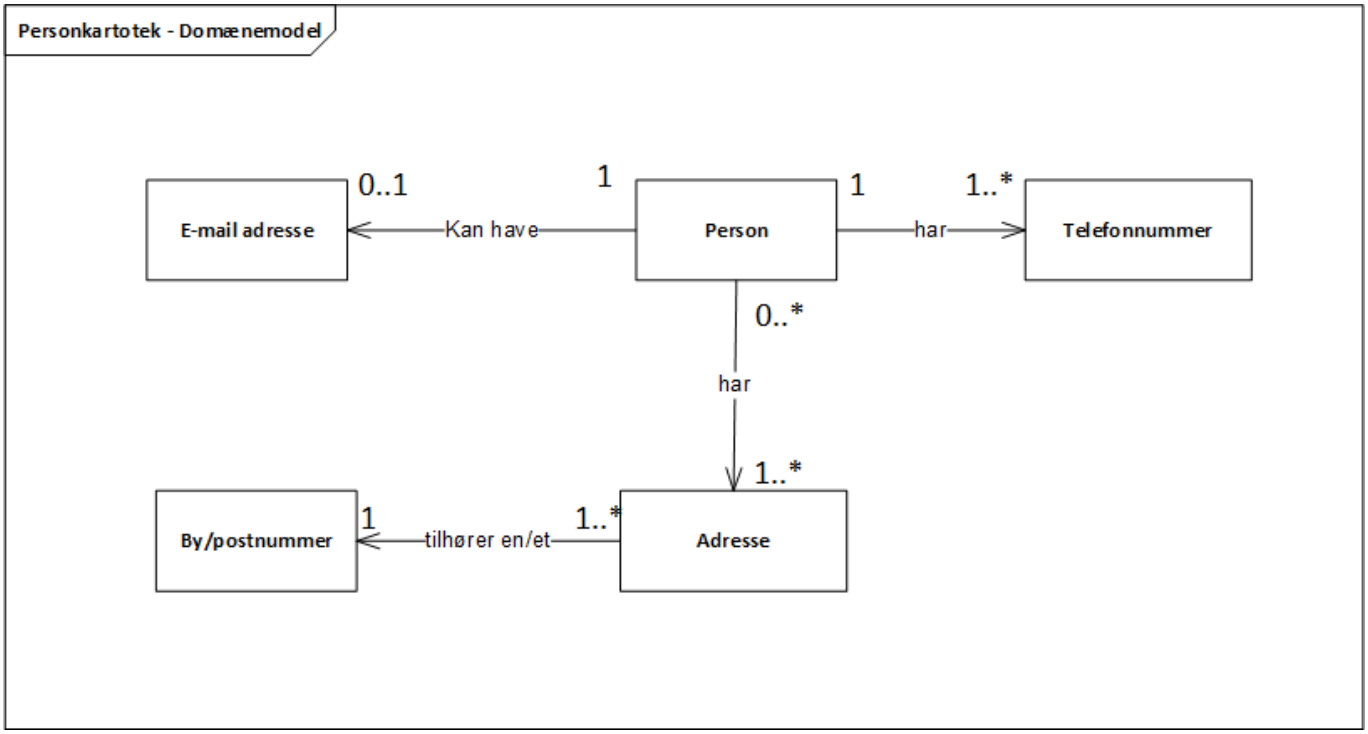

Figure 1: Domænemodellen udviklet i forbindelse med `Domain Driven Design`

2 Entity Relations diagram - ERD

Efter udarbejdelsen af domænemodellen opstilles et ERD diagram. (`Entity Relations Diagram`) Dette diagram ses på figur 2. Det bemærkes at ERD diagrammet har udeladt domæneklassen `Email adresse`, da denne er blevet en del af `Person` entity. Følgende relationer er blevet opstillet:

- 1 **Person** har M **Telefonnummer**
- M **Person** har M **Adresse**
- M **Adresse** tilhører 1 **By_Postnummer**

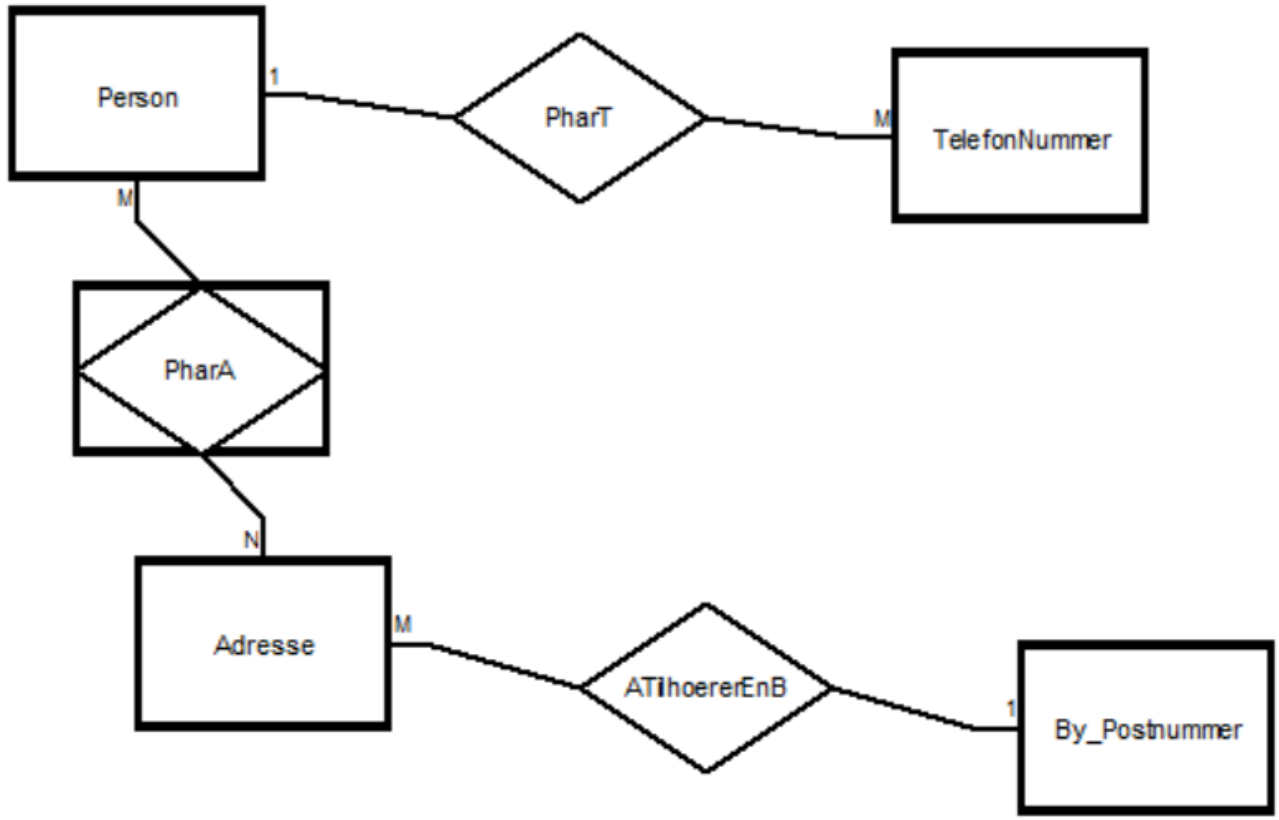

Figure 2: Domænemodellen udviklet i forbindelse med `Entity Relations Diagram`

3 Klassediagramer

Til løsning af denne opgave, er der udarbejdet to projekter til den relationelle database og 1 projekt til dokument databasen. De to projekter til den realtionelle database er bestående af et "Class library" og en konsolapplikation. Derimod er dokumentdatabasen kun bestående af en konsolapplikation. Et komplet klassediagram over klassebiblioteket til den relationelle database ses på figur 3.

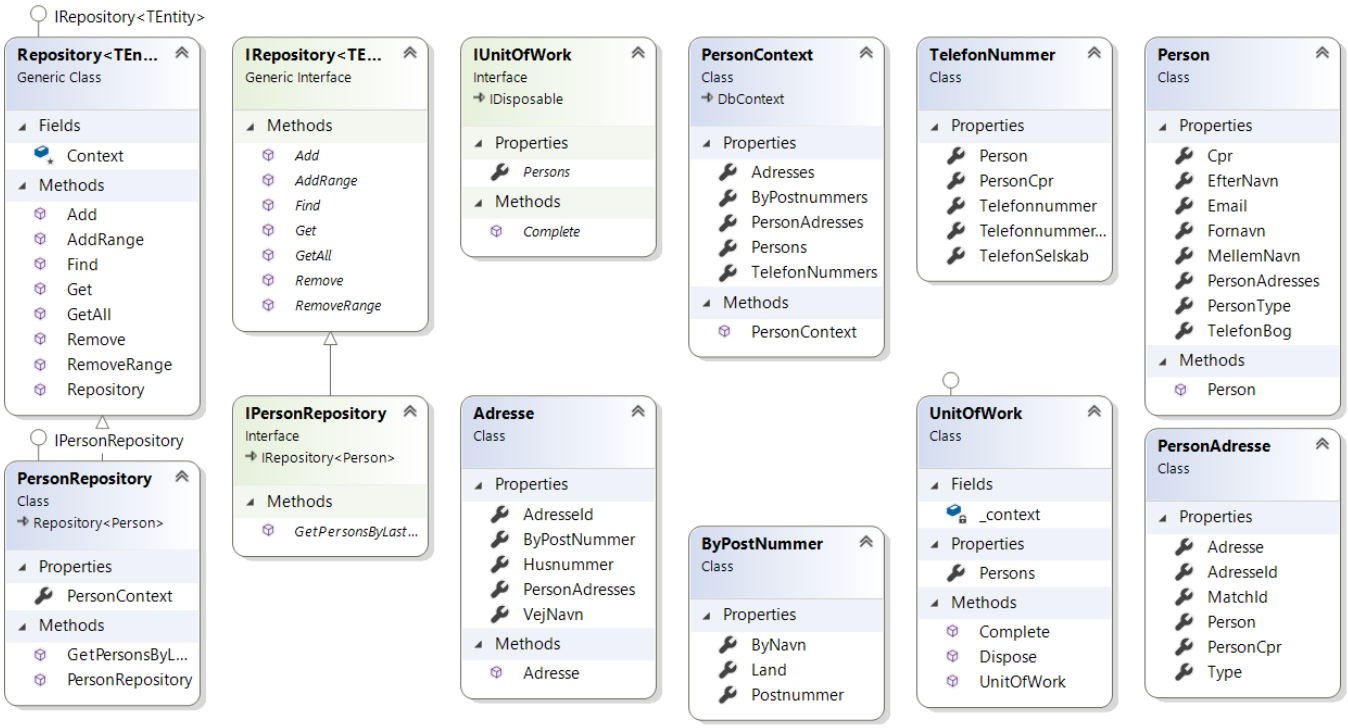

Figure 3: Klassediagram over den implementerede løsning til den relationelle database

Et komplet klassediagram over konsolapplikationen til dokument databasen ses på figur 4.

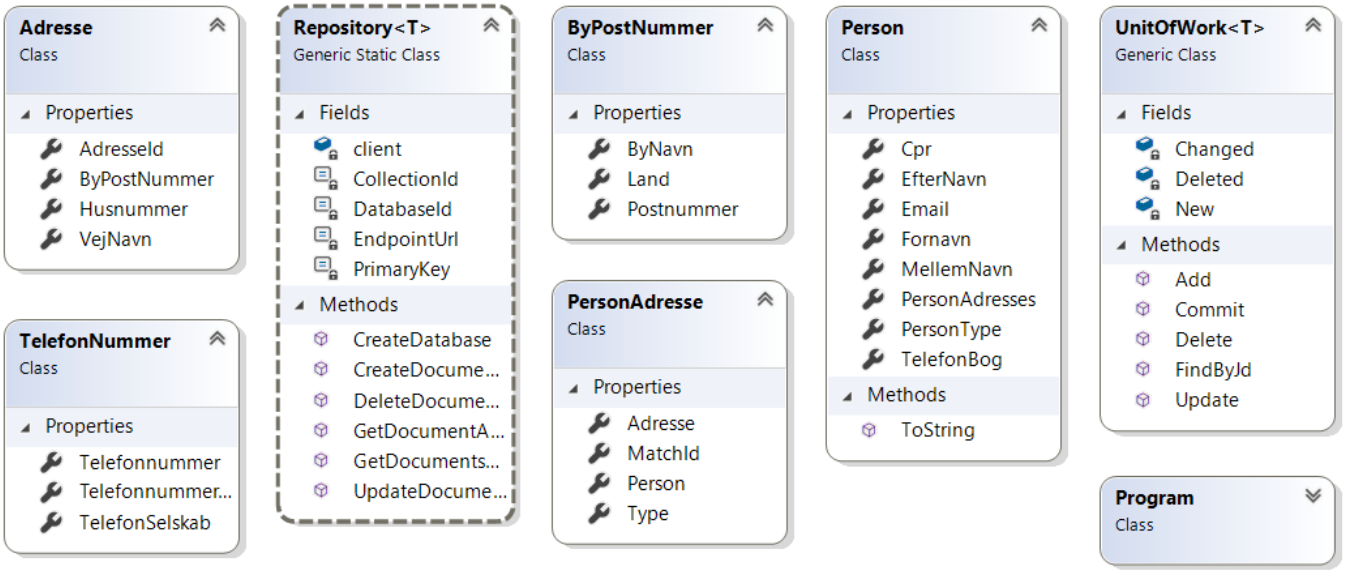

Figure 4: Klassediagram over den implementerede løsning til dokument databasen

4 Case Beskrivelser

Som en del af opgaven har vi skulle kigge nærmerer på datakravene, og dertil lave cases, for de bounded contexts. ud fra datakravene har vi lavet 5 cases, som vises på figurende herunder. figurende er desginet efter en træ struktur, hvis en knude ingen børn har er dette et field af sin forældre knude. hvis der fremkommer et undertræ af en knude er dette enten et value object eller en entity, dette vil blive beskrivet, ved forekomst.

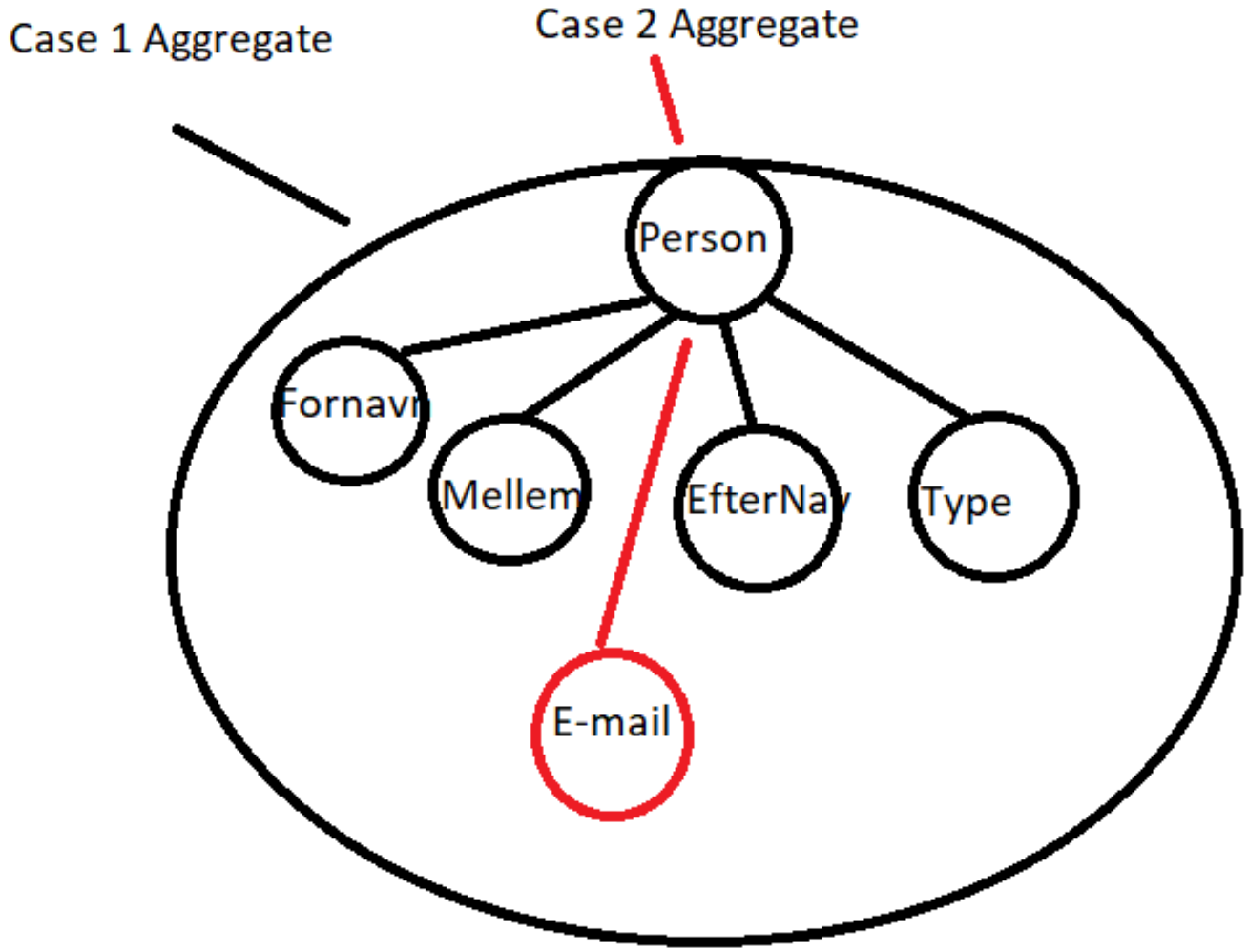

Figure 5: Case 1 og 2

Som det fremgå fra figur 5 ses case 1 og case 2, beskrivelsen for case 1 er som følger:

- Case 1: En persons fornavn, mellemnavn og efternavn samt typen af personen, hvor typen er afhængig af kontekst for personkartoteket. For denne context laver vi et design valg, som har en aggregate, med en person, som root entity. Vi definerer derefter efterfølgende de fields, som en person har, og viser at de kan tilgåes fra en person i denne case.
- Case 2: Hvis muligt en e-mailadresse til personen. for case 2. har vi samme aggregate, men beskrivelse er fokuseret på persons email. Dette er et seperat datakrav og har derfor også sin egen case. dette er igen identifiseret, som et field hos personen.

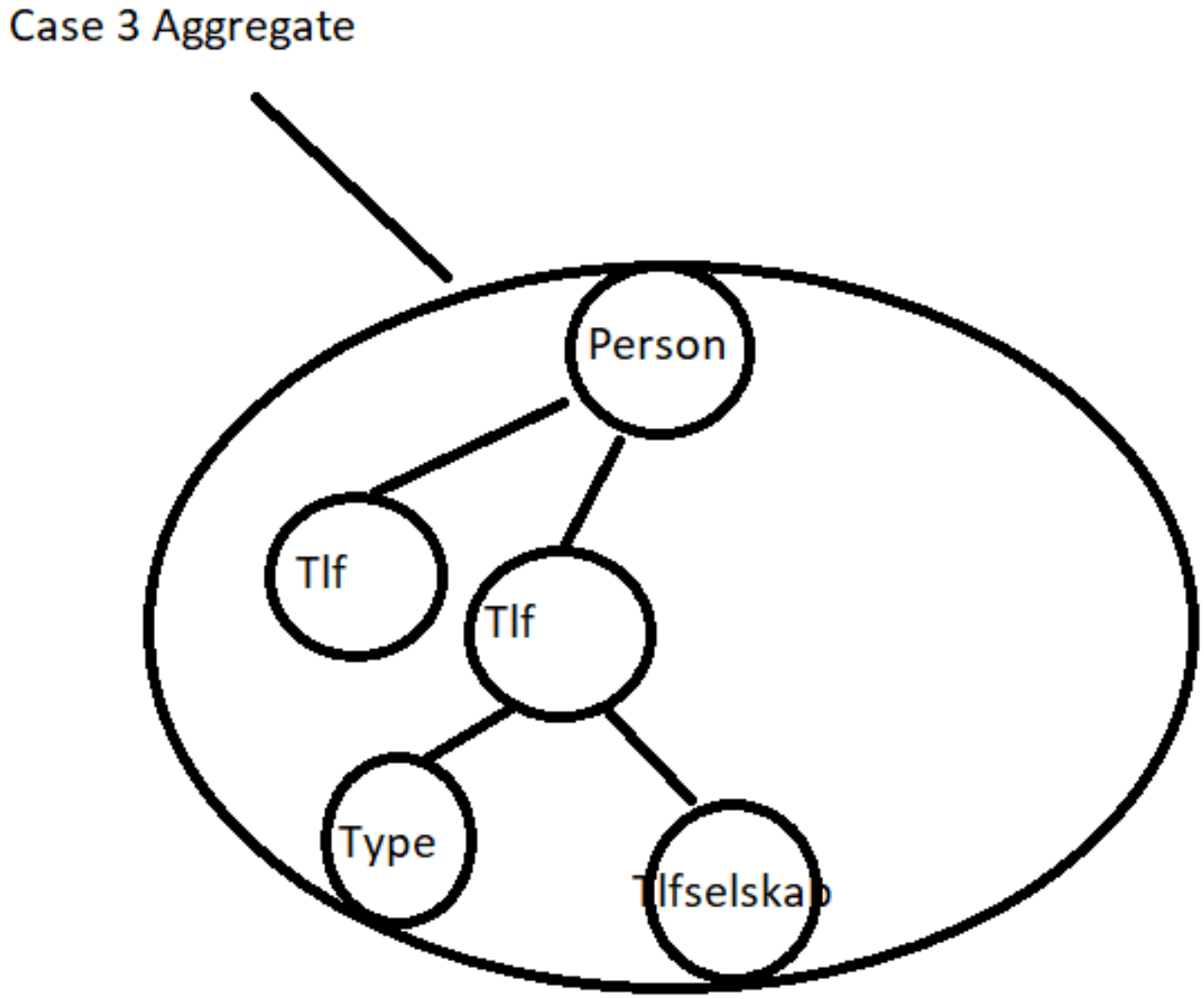

Figure 6: Case 3

Fra figur 6, ser vi case 3, hvor vi igen har taget et kig på datakravene og ser vores design valg herudfra, case er beskrivet således:

- Case 3: Et eller flere mulige telefonnumre til personen, samt oplysninger og telefonens brug (eks. privat, arbejde, mobil..) og kan det registreres: teleselskab bag telefonen (afht. afregning for opkald).
  - Vi har altså et aggregate, som igen anvender person som root, da dette er det bedste design valg udfra vores to tidligere cases, igennem person kan vi så tilgå en eller flere af personens telefonnr. dette er her tænkte eks. da vores applikation ikke afspejler dette, men designet giver mulighed for det. bemærk, at her er tlf. en entity, selv om dette ikke fremgå tydeligt fra figuren, så kan dette ses i koden da den har sin egen identifier, det gør også at man kan tilgå tlfs. fields, som det fremgå fra figuren, type og selskab.

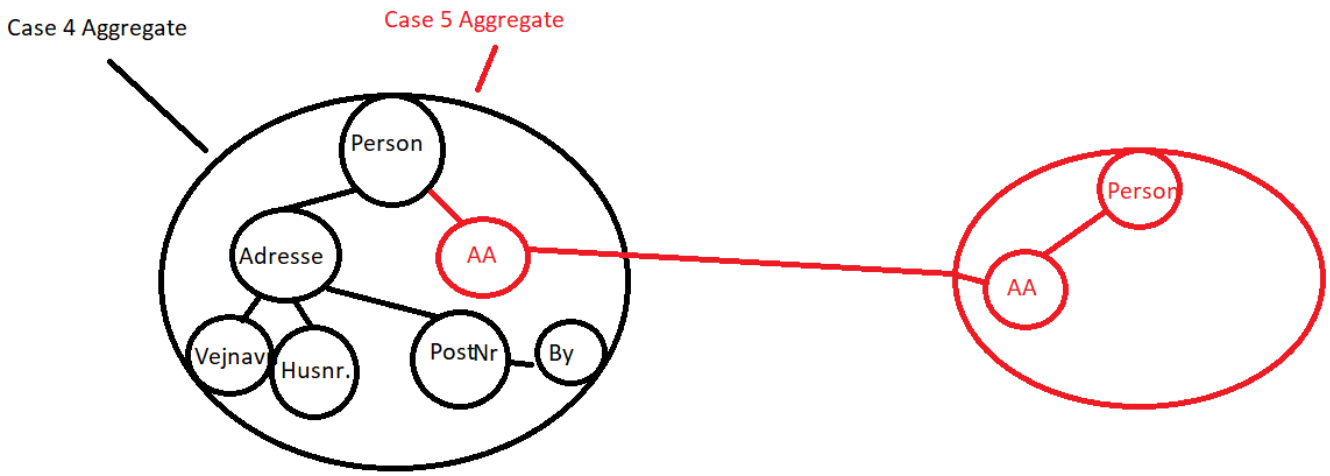

Figure 7: Case 4 og 5

Fra figur 7, ses vores 2 sidste cases lavet udfra datakravene, det er vigtigt at knytte en kommentar til at som om det ikke fremgår her så har postnummer også en tilhørende land field i sig, dette vil fremgå af koden, men er desværre ikke med her ved en fejl. Case beskrivelse 4, er som følger:

- Case 4: Adresseoplysninger: vejnavn og husnummer, postnummer og bynavn på personens primære kontaktadresse.
  - Her er ikke meget at knytte en kommentar til, det fremgår efter samme princip, som vores tidligere cases. det er værd at benærke at postNr. her er en entity. Men ser vi på case 5 er der her mulighed for diskussion:
- Case 5: Det er muligt for en person at have yderligere alternative adresser eks. privat, arbejde, sommerhus o. lign. Den samme adresse kan deles af flere personer, hvor de enkelte personer kan have forskellige adressetyper for samme adresse. Typen af hver persons tilknytning til adressen skal registreres. Typen af en alternativ adresse skal kunne registreres. Eks. Sommerhus, arbejde o.lign.
  - da flere personer er knyttet til den samme adresse, er der et væsentligt design valg, som skal tages, da en adresse hidtil har været en entity, har vi valgt at gøre det samme for alle typer af adresser, sådan at 2 personer har den samme entity, derfor skal entitien AA(alternativ adresse) i figur 7, ses, som det samme objekt knyttet til hver sin person.