Obligatorisk Oppgave 2 Databaser

Først starter vi med å grovt splitte opp kolonnene i regnearket inn i entiteter.





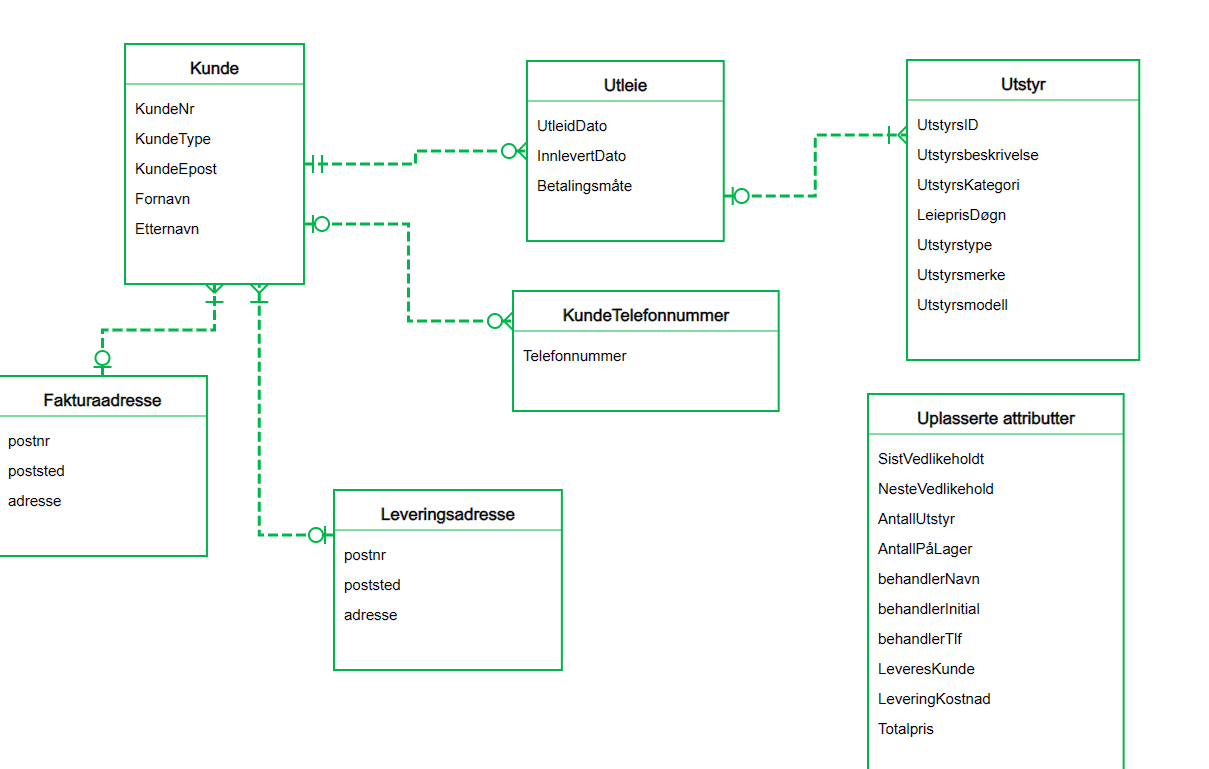
1. Normalisering av tabellen.

For å tilfredsstille kravene for 1. Normalform (1NF) må alle verdiene være atomære. At en verdi er atomær betyr at den ikke kan deles opp, eksempler på dette vil være lister med verdier. I tabellen vi tar utgangspunkt i ser vi flere slike kolonner hvor radene lagrer lister med verdier.





Vi velger derfor å dele disse opp i egne entiteter eller attributter.



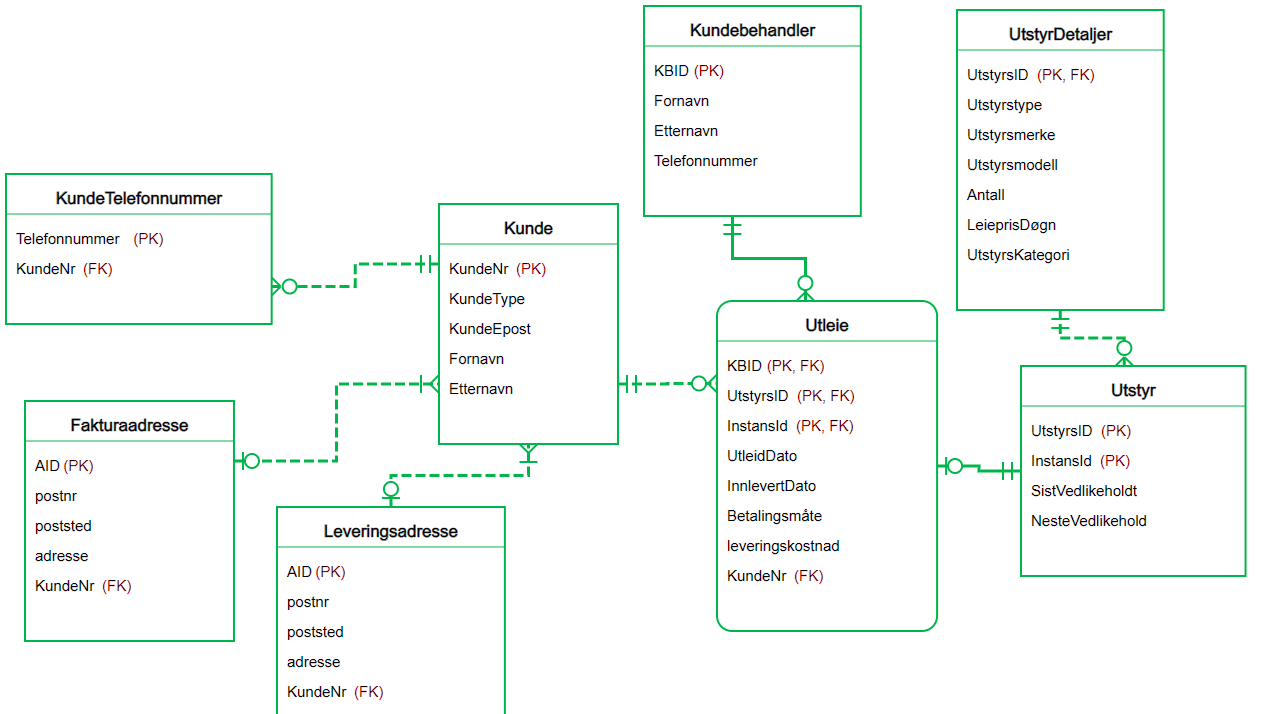
Vi har nå splittet attributtene inn i atomære attributter og flyttet enkelte inn i egne entiteter. Adressene har fått egne entiteter med postnr, poststed og adresse som attributter. Telefonnumrene til kunden har fått egen tabell som gjør det lettere å lagre flere telefonnummer per kunde. Til slutt har vi splittet opp et attributt i Utstyr til 3 attributter. Kundenavnet har blitt splittet opp til Fornavn og Etternavn, men siden dette også kan være navn på bedrifter endres det tilbake til bare navn i den siste versjonen. Behandler navn har derimot blitt splittet opp i senere tabeller.

Notat: Adressen kunne også ha blitt splittet opp i gate navn og gate nummer, men jeg oppdaget dette litt sent.

Modellen så langt er nå på 1NF og vi kan gå videre herfra.

Merk at forholdene og primærnøkklene er enda ikke justert vi er fortsatt på konseptuell fase. I tillegg er ikke alle attributtene plassert.

For å gå videre med å verifisere 2NF trenger vi å vite primærnøkklene til entitetene våre. Vi fullfører derfor forholdene og plasserer ut de siste attributtene.



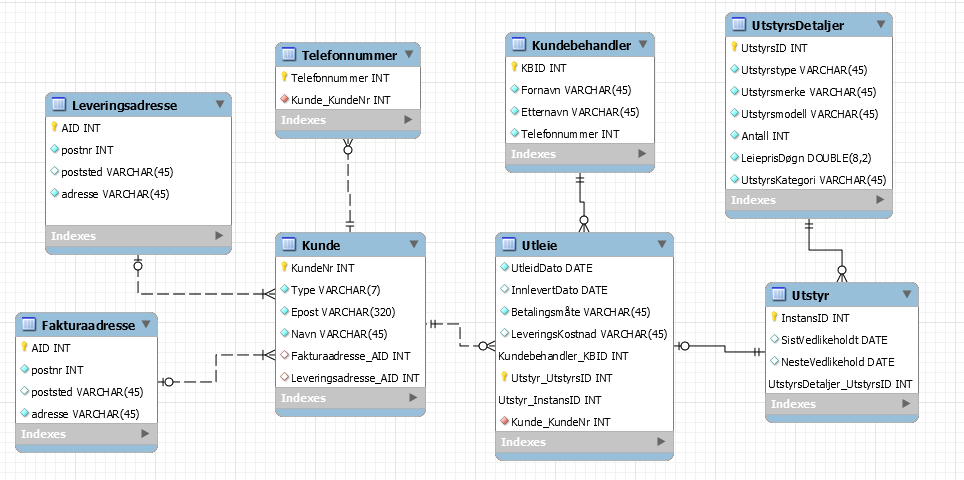
Tabellen vår tilfredsstiller 2NF hvis den ikke har attributter med partiell avhengighet av en primærnøkkel. Dette kunne vi tidligere se i entiteten Utstyr. Her er primærnøkklene UtstyrsId og InstansId begge disse lar oss skille mellom hvert enkelt utstyr selv om de er av samme sort. Avhengigheten her kommer av at alle detaljene som nå er i entiteten «UtstyrDetaljer» var avhengig av bare UtstyrsID. Derfor valgte vi å lage en ny entitet med info om et utstyr basert på bare UtstyrsID.



3NF tilfredsstilles hvis en tabell ikke inneholder transitive funksjonelle avhengigheter. Dette vil si at hvis et attributt som ikke er en primærnøkkel er bestemt av en annen attributt som heller ikke er en primærnøkkel. Vi finner et slikt tilfelle i adresse tabellene våre. Her lagres både postnr og posted, men postnr kan bli brukt til å finne ut poststedet. Hva vi kunne ha gjort for å rette på dette var å lage en til tabell som lister postnumre og korresponderende poststeder, men vi valgte å ikke gjøre dette ettersom det kunne bli unødvendig med rot. Med mindre en allerede har en liste med postnr og korresponderende poststed vil poststedet være brukerdefinert som utsetter det for skrivefeil allikevel.

Boyce-Codd NormalForm tilfredsstilles selvfølgelig ikke når vi ikke følger 3NF. Her støter vi på de samme problemene som i 3NF med at postnr er en determinant.

Nå er det på tide å føre dette inn i mysql workbench. Vi vil nok måtte gjøre endringer på enkelte forhold samt må vi innføre datatypene.



Endringer ble gjort på forholdene mellom Utstyrsdetaljer og Utstyr. Utstyr er nå en svak entitet som arver primærnøkkelen fra UtstyrsDetaljer. Fremmednøklene er også flyttet inn i kunde i stedet for å ha dem i adressene selv. Dette var fordi en adresse kunne ha flere kunder, mens motsatt ikke kunne skje.

SQL Spørringene:

#A

select KundeNr, Type, Navn, Epost, Telefonnummer , Fakturaadresse.adresse as Fakturaadresse, Fakturaadresse.postnr as FakturaPostnr, Leveringsadresse.adresse as Leveringsadresse, Leveringsadresse.postnr as LeveringsPostNr from Kunde inner join Fakturaadresse on Fakturaadresse\_AID = Fakturaadresse.AID inner join Leveringsadresse on Leveringsadresse\_AID = Leveringsadresse.AID inner join Telefonnummer on Kunde\_KundeNr = KundeNr where Type = "Bedrift";

#B

create or replace view KunderMedFaktura as select KundeNr, Navn, postnr, poststed, adresse from Kunde, Fakturaadresse, Utleie where KundeNr = Kunde\_KundeNr and Fakturaadresse\_AID = AID group by Kunde\_KundeNr;

create or replace view FlestOrdre as select Kunde\_KundeNr,count(\*) as Ordre from Utleie group by Kunde\_KundeNr having count(\*) =(select max(Ordre) from (select Kunde\_KundeNr,count(\*) as Ordre from Utleie group by Kunde\_KundeNr) Utleie);

select Navn, postnr, poststed, adresse from FlestOrdre inner join KunderMedFaktura on Kunde\_KundeNr = KundeNr;

#C

create or replace view KundenMestPenger as select sum(-1\*datediff(UtleidDato, InnlevertDato) \* LeieprisDøgn + LeveringsKostnad) as "Tjent per utstyr",Kunde\_KundeNr, Utstyrsmerke, Utstyrsmodell, Utstyrstype from Utleie inner join UtstyrsDetaljer on Utstyr\_UtstyrsID = UtstyrsID where InnlevertDato is not null group by Kunde\_KundeNr;

select \* from KundenMestPenger inner join Kunde on Kunde\_KundeNr = KundeNr inner join Leveringsadresse on Leveringsadresse\_AID = AID where `Tjent per utstyr` = (select max(`Tjent per utstyr`) from KundenMestPenger);

#D

create or replace view VanligsteUtstyr as select Utstyr\_UtstyrsID, count(\*) as ganger from Utleie group by Utstyr\_UtstyrsID;

select MAX(ganger) as "ganger utleid", Utstyrsmerke, Utstyrsmodell, Utstyrstype, UtstyrsKategori from VanligsteUtstyr inner join UtstyrsDetaljer on Utstyr\_UtstyrsID = UtstyrsID;

#E

select sum(-1\*datediff(UtleidDato, InnlevertDato) \* LeieprisDøgn + LeveringsKostnad) as "Tjent per utstyr", Utstyrsmerke, Utstyrsmodell, Utstyrstype from Utleie inner join UtstyrsDetaljer on Utstyr\_UtstyrsID = UtstyrsID where InnlevertDato is not null group by UtstyrsID order by -1\*datediff(UtleidDato, InnlevertDato) \* LeieprisDøgn + LeveringsKostnad desc;

#F

SELECT \* FROM Utleie inner join Kundebehandler on Kundebehandler\_KBID = KBID where InnlevertDato is null and Fornavn = "Hilde" and Etternavn = "Pettersen";