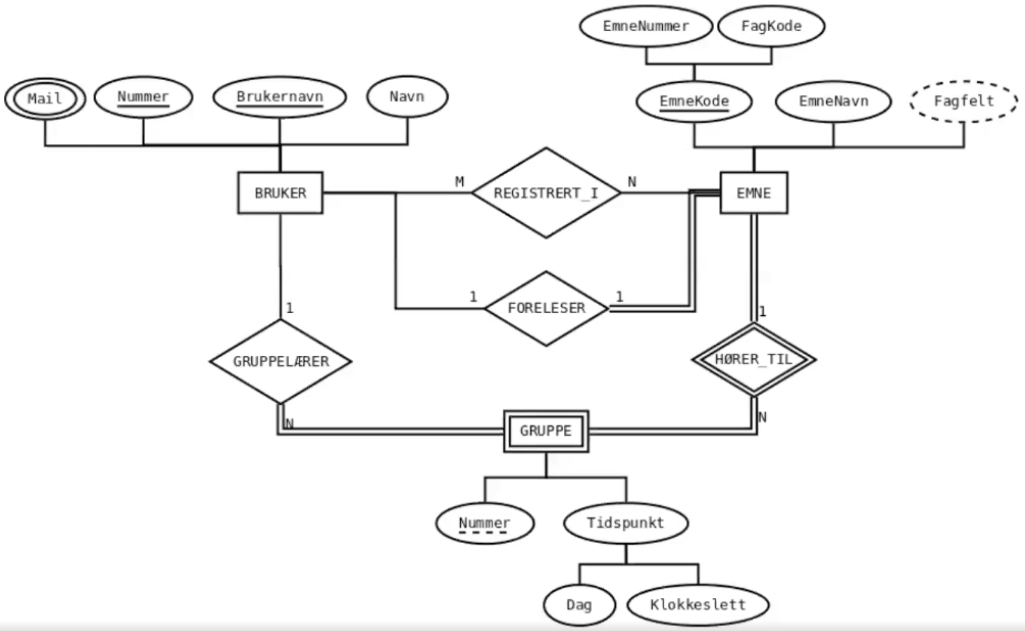
Realiser følgende ER-diagram til et databaseskjema.



Ved realisering så har vi en realiseringsalgoritme som følges når vi skal realisere et gitt ER-diagram til et databaseskjema. **I eksamen så er det utrolig viktig at man noterer seg valgene man gjør ved realisering av et gitt databaseskjema. Ellers kan man miste en del poeng så husk dette !!!**

Steg 1: Start med de enkel entitetene, noter deres attributter og eventuelle kandidat og primærnøkler.

Ved primærnøkkelen så skal man velge “1” av kandidatnøklene så vi kan velge “Nummer”. Kunne også ha valgt “Brukernavn”, men kan være lurt å velge det mest fornuftige hvor i dette tilfellet så er “Nummer” mer unikt enn “Brukernavnet”. Viktigste er å være “1” av kandidatnøklene.

Bruker(Nummer, Brukernavn, Navn)
KN:{Nummer, Brukernavn}
PN: {Nummer}

Vi har at BRUKER-entiteten har en multi-verdi(“Mail”) men disse utsettes til senere som står i realiseringsalgoritmen siden det er egne regler for slike verdier. Vi kan realisere “EMNET”-entiteten som neste.

Før vi starter så er det viktig å notere at “EMNE” har en såkalt “Composite attributt”. Dvs, en attributt som er avhengig av flere attributter hvor i dette tilfellet tilsvarer “Emnekode”. Denne er da avhengig av “Emnenummer” og “Fagkode”. Vi skal kun notere de attributtene som utgjør “Composite attributtet”, så “Emnenummer” og “Fagkode” trengs å noteres, ikke “Emnekode”. I tillegg skal ikke “Fagfelt” inkluderes ettersom den er utledbar og disse attributtene er ikke nødvendig å notere ettersom “Fagkode” hjelper oss med å identifisere et gitt fagfelt. I tillegg til at det blir en form for dobbeltlagring hvor vi lagrer flere ganger, noe som er unødvendig. Da kan vi starte å realisere denne relasjonen:

Emne(Emnenummer, Fagkode, Emnenavn)
KN: {Emnenummer, Fagkode}
PN: {Emnenummer, Fagkode}

Her utgjør “Emnenummer” og “Fagkode” kandidat- og primærnøkkelen vår siden disse to utgjorde unikheten for en gitt emne. Dette kan sees gjennom modellen over ettersom disse to attributtene utgjorde nøkkelen “Emnekode”, så dermed er disse attributtene både kandidat- og primærnøkkel.

Steg 2: Realiser de svake entitetene, noter deres attributter og fremmednøkler.

Ved realisering av de svake entitetene så må vi huske at vi må inkludere de svake nøklene og nøkkelen til den gitte entiteten som gjør den svake identiteten “Identifiserbar” eller får den unike karakteristikken. Dermed må “GRUPPE”-entiteten ha primærnøkkelen til “EMNE”-entiteten i sin relasjon. Dermed inkluderer vi primærnøkkelen til “EMNE” som tilsvarer da både {Emnenummer, Fagkode} og “GRUPPE”-entiteten inkludert.

Gruppe(Nummer, Dag, Klokkeslett, Emnenummer, Fagkode)
KN: {Emnenummer, Fagkode, Nummer}
PN: {Emnenummer, Fagkode, Nummer}

Vi må også nevne at “GRUPPE”-entiteten har fremmednøklene som er da {Emnenummer, Fagkode} så det gjør vi slikt:
Fremmednøkler:
Gruppe(Emnenummer, Fagkode) -> Emne(Emnenummer, Fagkode)

Steg 3: Realiser de ulike relasjonene som knytter to entiteter med hverandre
Først starter vi med å realisere de relasjonene som har 1:1(en til en) i kardinalitet. Deretter tar vi de som er 1:N(en til mange) og tilslutt N:M(Mange til mange) relasjonene.

De ulike alternativene som Leif forklarte:

1. Legg inn attributter i den ene entitetens (database) relasjon som er fremmednøkkel til den andre (slik som gjort i løsningsforslaget, og forklart mer i svaret mitt under).

2. Lag én relasjon som representerer begge entitetene òg relasjonen mellom dem. Altså at både `EMNE`, `BRUKER` og relasjonen `FORELESER` blir representert med én tabell. Dette er kun ønskelig hvor entitetene egentlig kan ansees for å være del av samme entitet, noe som er litt sjeldent og derfor ikke bruker særlig ofte. Det er heller ikke alltid mulig, dersom hver av de to entitetene har relasjoner til andre entiteter.

3. Lag en helt ny (database) relasjon som inneholder attributter som er fremmednøkler til de to entitetenes primærnøkler (det ville her blitt `Foreleser(bruker, emnenummer, fagkode)` hvor `Foreleser(bruker) -> Bruker(nummer)` og `Foreleser(emnenummer, fagkode) -> Emne(emnenummer, fagkode)`). For 1-1-relasjoner bør dette kun gjøres dersom det er få relasjoner mellom de to entitetene. F.eks. om vi har entitetene `PERSON` og `DYR` og en relasjon `EIER`, hvor vi antar at en person kun kan eie ett dyr, og at et dyr kan kun eies av én person. La oss si at databasen her skal inneholde alle mennesker og dyr i hele verden. Det er jo veldig mange dyr som ikke eies av noen, og veldig mange mennesker som ikke eier noe dyr. Om man da bruker en fremmednøkkel (altså alternativ 1. over) vil man ende opp med mange `NULL` verdier, hver gang en person ikke eier et dyr (eller motsatt). Da kan denne varianten være lur å bruke.

1.) Du trenger kun å inkludere den valgte primærnøkkelen (altså den av `nummer` eller `brukernavn` som man i realiseringen av `BRUKER` valgte som primærnøkkel) i `EMNE`. Så siden jeg valgte `nummer` til å være `Bruker`s primærnøkkel vil vi legge inn en fremmednøkkel til denne i `Emne` (jeg valgte å kalle dette attributtet i `Emne` for `foreleser`, men man kunne jo også kalt det noe annet om man vil). Så da får vi `Emne(foreleser) -> Bruker(nummer)`. Du skal ikke ta med noe annet, heller ikke

andre attributter i `Bruker` som er fremmednøkler tilbake til `Emne`. Svarte det på spørsmålet?

2.) Som sagt over er det oftest man bruker alternativ 1., med mindre det er få elementer som er relatert med relasjoner, hvor alternativ 3 er lurt. Alternativ 2. er som sagt sjeldent man bruker. Svarte det på spørsmålet ditt?

Steg 3A: Realisere de relasjonene som har 1:1

Ved realisering av slike relasjoner så finnes det "3" valg når vi skal realisere de ulike relasjonene. Dermed starter vi å realisere "FORELESER"-relasjonen som knytter da BRUKER og EMNE. Bruker da valg 3.).

1.) Lag en ny realisering som representerer relasjonen og inkluder kandidatnøkkelene fra begge entiteten som knyttes til den gitte relasjonen. Noterer også primær- og kandidatnøklene for den nye relasjonen. Lag en helt ny (database) relasjon som inneholder attributter som er fremmednøkler til de to entitetenes primærnøkler
Gjennom "FORELESER"-relasjonen så får vi da følgende:
Foreleser(Emnenummer, Fagkode, Nummer)
KN: {Emnenummer, Fagkode, Nummer}
PN: {Emnenummer, Fagkode, Nummer}

2.) Lag én relasjon som representerer begge entitetene òg relasjonen mellom dem. Altså at både `EMNE`, `BRUKER` og relasjonen `FORELESER` blir representert med én tabell. Dette er kun ønskelig hvor entitetene egentlig kan ansees for å være del av samme entitet, noe som er litt sjeldent og derfor ikke bruker særlig ofte. Det er heller ikke alltid mulig, dersom hver av de to entitetene har relasjoner til andre entiteter. Ikke ønskelig å bruke denne, sjeldent den benyttes i følge Leif.

3.) Legg inn attributter i den ene entitetens (database) relasjon som er fremmednøkkel til den andre

Du trenger kun å inkludere den valgte primærnøkkelen (altså den av `nummer` eller `brukernavn` som man i realiseringen av `BRUKER` valgte som primærnøkkel) i `EMNE`. Så siden jeg valgte `nummer` til å være `Bruker`s primærnøkkel vil vi legge inn en fremmednøkkel til denne i `Emne` (jeg valgte å kalle dette attributtet i `Emne` for

`foreleser`, men man kunne jo også kalt det noe annet om man vil). Så da får vi `Emne(foreleser) -> Bruker(nummer)`. Du skal ikke ta med noe annet, heller ikke andre attributter i `Bruker` som er fremmednøkler tilbake til `Emne`.
I dette tilfellet, er det best å sette "Foreleser"-attributtet inn i EMNE-relasjonen siden den har "total utledning" i relasjonen så dermed er dette mest fornuftig ettersom EMNE-relasjonen må minst inkludere en "Foreleser" i relasjonen. Dermed får vi følgende:

Emne(Emnenummer, Fagkode, Emnenavn, Foreleser)
Ved å gjøre dette så må du sette fremmednøkkelen for det nye attributtet og sette det mot primærnøkkelen til det andre relasjonen. Så vi får da følgende:

Fremmednøkler
Emne(Foreleser) -> Bruker(Nummer)

Steg 3B: Realisere de relasjonene som har 1:N
Ved realisering av slike relasjoner så finnes det "2" valg når vi skal realisere de ulike relasjonene.

1.) Lag en ny realisering som representerer relasjonen og inkluder kandidatnøkkelene fra begge entiteten som knyttes til den gitte relasjonen.

2.) Legg inn attributter i den ene entitetens (database) relasjon som er fremmednøkkel til den andre

Velger nummer 2.) og får da følgende:
Gruppe(Nummer, Dag, Klokkeslett, Emnenummer, Fagkode, Gruppelærer)

Fremmednøkler:
Gruppe(Gruppelærer) -> Bruker(Nummer)

Steg 3C: Realisere de relasjonene som har N:M
Ved realisering av slike relasjoner så finnes det "1" valg når vi skal realisere de ulike relasjonene.

1.) Lag en ny realisering som representerer relasjonen og inkluder kandidatnøkkelene fra begge entiteten som knyttes til den gitte relasjonen.

    Dette er greit ettersom vi kun har dette valget tilgjengelig. Ved kardinaliteter som ikke tilsvarer “N:M”, men “1:M” eller “1:1”, så brukes ellers ikke dette valget.

Siden vi kun har et valg, så bruker vi den og får da følgende:
Registrert_i(Nummer, Emnenummer, Fagkode)
KN: {Nummer, Emnenummer, Fagkode}
PN: {Nummer, Emnenummer, Fagkode}

Fremmednøkler:
Registrert_i(Nummer) -> Bruker(Nummer)
Registrert_i(Emnenummer, Fagkode) -> Emne(Emnenummer, Fagkode)

Steg 4: Realiser multiverdiene
Lager en ny relasjon som uttrykker multiverdiene, i dette tilfellet vil det være “Mail” og inkludere kandidatnøklene som inkluderes for entiteten som “Mail” tilhører som er da “BRUKER”. Dermed får vi følgende:

Mail(mail, nummer)
KN: {mail, nummer}
PN: {mail, nummer}

Fremmednøkler:
Mail(nummer) -> Bruker(nummer)

Slik ser hele realiseringen ut:

```
# Relasjoner

Bruker(nummer, brukernavn, navn)
- KN: {nummer}, {brukernavn}
- PN: {nummer}
Emne(emnenummer, fagkode, emnenavn, foreleser)
- KN: {emnenummer, fagkode}
- PN: {emnenummer, fagkode}
Gruppe(nummer, emnenummer, fagkode, dag, klokkeslett, gruppelærer)
- KN: {nummer, emnenummer, fagkode}
- PN: {nummer, emnenummer, fagkode}
Registrert_i(nummer, emnenummer, fagkode)
- KN: {nummer, emnenummer, fagkode}
- PN: {nummer, emnenummer, fagkode}
Mail(nummer, mailadresse)
- KN: {nummer, mailadresse}
- PN: {nummer, mailadresse}

# Fremmednøkler

Gruppe(emnenummer, fagkode) -> Emne(emnenummer, fagkode)
Emne(foreleser) -> Bruker(nummer)
Gruppe(gruppelærer) -> Bruker(nummer)
Registrert_i(nummer) -> Bruker(nummer)
Registrert_i(emnenummer, fagkode) -> Emne(emnenummer, fagkode)
Mail(nummer) -> Bruker(nummer)
```