Øving 8

Problemstilling:

Gruseløkka linjeforening ønsker et medlemskap system for å lagre brukere som skal være medlem av forskjellige grupper som trinn, klasser, komiteer og undergrupper i linjeforeningen. En gruppe skal ha et navn, beskrivelse, kontakt email og type. En gruppe kan ha flere tillatelser, og en tillatelse kan ha flere grupper.

En tillatelse sier hva man kan gjøre, og består av en id og et enum som for eks. "lag_nyhet".

En bruker i linjeforeningen skal kunne melde seg inn flere grupper. Systemet skal lagre brukernavn og navn til brukeren. Medlemskap skal også lagres i systemet, med start-dato og type. Type er for eksempel leder, nestleder, medlem osv. Når en brukers medlemskap er over, skal dette også lagres i databasen med den reelle slutt datoen.

Grunnen til at et medlem trenger tillatelser gjennom å være medlem i en gruppe, er for å vite om han/hun skal ha tilgang til å gjøre diverse ting gjennom systemet. I denne problemstillingen har vi kun lagt til nyheter, men man kunne utvidet dette med annen funksjonalitet uten nye tabeller for tillatelser.

Relasjonsmodellen

```
gruppe(gruppe_id,navn, beskrivelse, kontakt_email, type)

gruppe_tillatelser(tillatelse_id, gruppe_id*, tillatelse_id)

tillatelse(tillatelse_id, beskrivelse)

bruker(bruker_id, brukernavn, fornavn, etternavn,email)

medlemskap(bruker_id*, gruppe_id*, start_dato, type)

medlemskap_historie(bruker_id*, gruppe_id*, start_dato, slutt_dato, type)

nyheter(nyhet_id, tittel, innhold, bruker_id*, laget_dato)
```

SQL-spørringer

Hent ut nyheter og brukeren som har skrevet den

```
SELECT nyhet.*, bruker.fornavn, bruker.etternavn
    FROM nyheter n
    JOIN bruker b ON n.bruker.id = b.bruker_id;
```

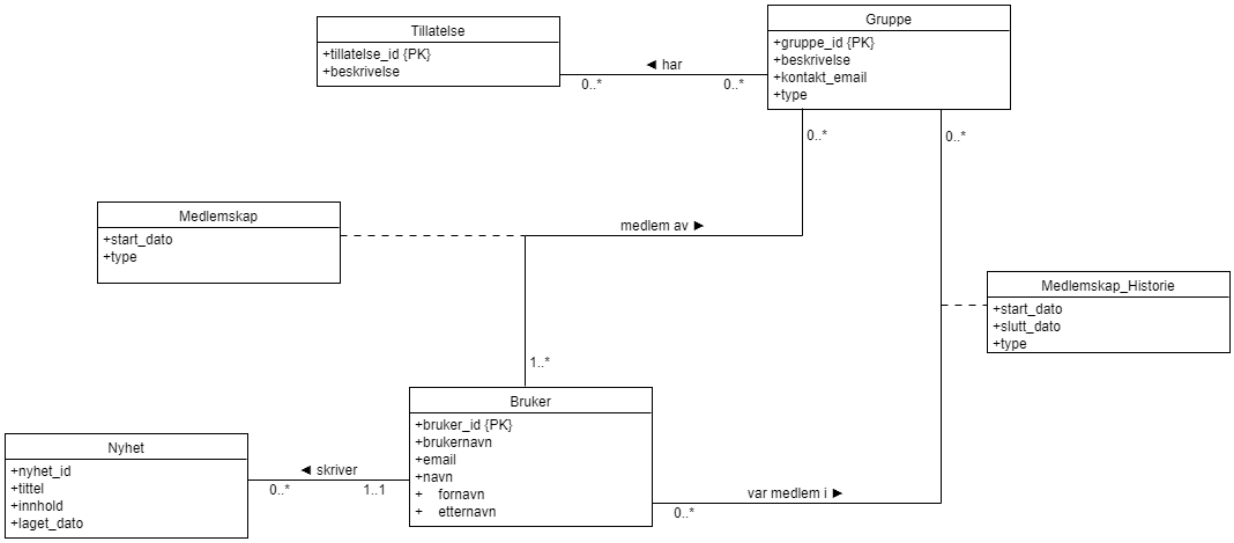
Hente ut en tillatelse for en spesifikk bruker

```
SELECT tillatelse.beskrivelse, DISTINCT(tillatelse_id), bruker.brukernavn
  FROM bruker
  JOIN medlemskap ON (bruker.bruker_id = medlemskap.bruker_id AND
bruker.bruker_id  = 4(vilkårlig valgt bruker_Id))
  JOIN gruppe ON (medlemskap.gruppe_id = gruppe.gruppe_id)
  JOIN gruppe_tillatelse ON (gruppe.gruppe_id =gruppe_tillatelse.gruppe_id)
  JOIN tillatelse ON (gruppe_tillatelse.tillatelse_id =
tillatelse.tillatelse_id  AND tillatelse.tillatelse_id = 3(vilkårlig valgt
tillatelse_id));
```

Hente medlemhistorikk til en bruker

```
SELECT medlemskap_historie.*
  FROM bruker
  JOIN medlemskap_historie ON (bruker.bruker_id =
medlemskap_historie.bruker_id);
```

EER-Diagram



Erfaring

Vi brukte MySQL som databaseløsning. Har logisk og enkle setninger som gjør det å hente data veldig enkelt. Kan bli lange og veldig nøstede setning, dersom du skal filtrere og legge sammen tabeller. Som sagt i problemstilling kan vi legge til mer arbeid en bruker kan gjøre med de forskjellige tillatelse nivåer.

XML

I likhet med SQL så formaterer XML data innhold på en svært strukturert måte, og blir likeså enkelt å få gjennom spørring som SQL. Hvis dataen er svært uregelmessig, som at en tabell inneholder mange null verdier, og kan da kanskje ha flere kolonner enn nødvendig, kan en delvis XML løsning være bra, noe som ikke er tilfelle i vårt case hvor null verdier oppstår sjeldent. En fullstendig XML løsning er hovedsakelig nødvendig dersom man skal lagre strukturer som er svært variable og er vanskelig å få inn i fast struktur, som da lagring av tekstdokumenter, vi bruker tekst i tabellene, som da er fordelt inn i tittel og innhold, noe vi tenker ikke trenger å fordeles inn i et XML dokument, siden vi tenker heller å lagre i rå tekst uten spørre funksjon eller navigasjons funksjoner. Et annet problem med XML er at filene er svært store sammenlignet med SQL løsninger. I motsetning til SQL system, trenger ikke en XML løsning å følge et fast skjema av attributter. XML gir også muligheter for metadata, altså at vi kan lagre beskrivelse for data. Tillater også å lagre en type tre-struktur.