IN2090 – Prøveeksamen 2019

Oppgaven består av tre deler, en om modellering, en om SQL og en om relasjonsmodellen. Poengene i parentes sier maksimalt antall poeng for oppgaven/delen. Det kan være lurt å tegne diagrammene i modelleringsoppgavene for hånd for å forberede seg på eksamen, samt løse SQL oppgavene uten å lage noen eksempeldatabase.

Modellering (30)

Oppgave 1 (20)

I denne oppgaven skal vi lage en ER-modell som beskriver land, regioner og byer. Tegn tydelig. Modellen skal inneholde følgende informasjon:

1. Et land har et unikt navn, en mengde offisielle språk, og en valuta. I tillegg har land et innbygertall som kan utledes fra dens byers inbygertall (se under).

2. En region har et unikt navn.

3. En region er inneholdt i nøyaktig ett land, og en by tilhører nøyaktig én region. Et land må inneholde minst én region, men kan inneholde mange. En region må ha minst en tilhørende by, men kan ha mange.

4. En by har et navn som kun er unikt innad i regionen den tilhører. For å unikt kunne identifisere en by må vi vite både dens navn og regionen den tilhører sitt navn.

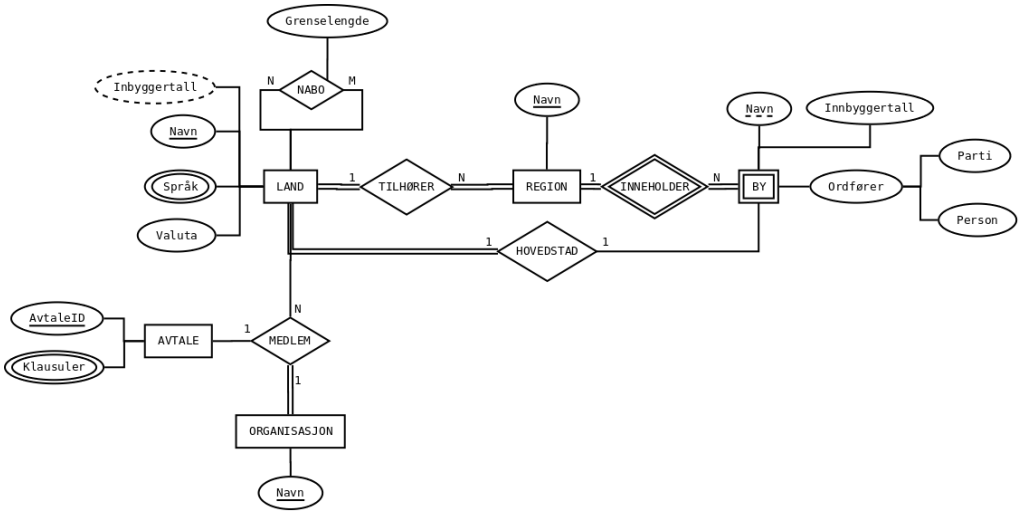
5. Hvert land har nøyaktig én by som hovedstad. En by kan kun være hovedstad for ett land.

6. En by har i tillegg til sitt navn (nevt over), et innbyggertall og en ordfører bestående av et partinavn og personnavn (f.eks. er ordføreren i Oslo ("SV", "Marianne Borgen")).

7. Et land kan ha mange andre land som naboland og et land kan være nabo til mange land. Når to land er naboer er det tilknyttet en grenselengde (altså lengden på grensen mellom de to landene) til dette naboskapet. F.eks. kan vi si at Norge er nabo med Sverige med en grenselengde på 1630km.

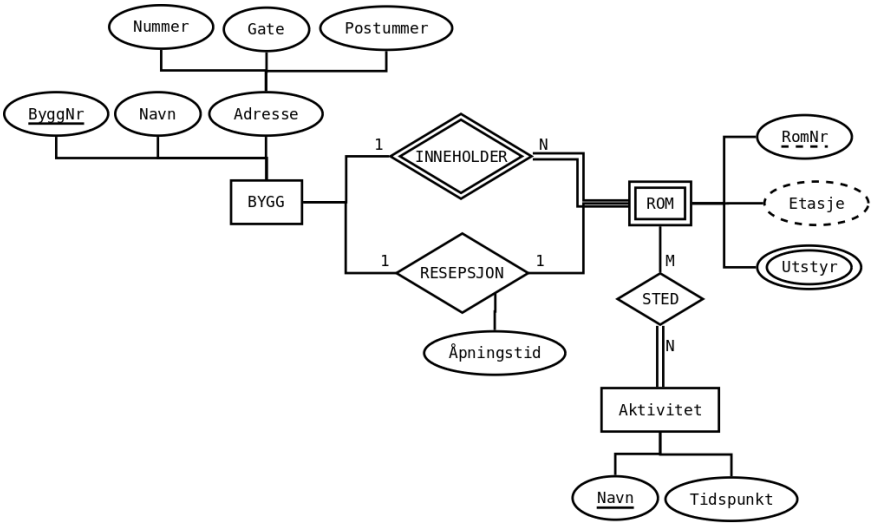
8. Et land kan være medlem i en organisasjon under en avtale (hint: Bruk en ternær relasjon). En organisasjon er unikt identifisert av et navn (f.eks. "EU" eller "NATO"). En avtale er unikt identifisert av en avtale-ID og har i tellegg en mengde klausuler. Et land kan, med en bestemt avtale, kun være med i en organisasjon; et land kan kun ha én avtale med en bestemt organisasjon; men en organisasjon kan ha mange medlemsland med samme avtale. I tillegg må en organisasjon ha minst ett medlemsland med en avtale.

Løsningsforslag



Oppgave 2 (10)

Realiser følgende ER-diagram til et databaseskjema:



Bruk understreking for å markere kandidatnøkler, og bruk i tillegg **fet skrift** for den valgte primærnøkkelen. List i tillegg opp alle fremmednøkler på formen $T(A) \rightarrow P(B)$ (her er attributt(ene) A i T en fremmednøkkel som refererer til attributt(ene) B i P). Skriv også opp eventuelle valg du tar underveis.

Løsningsforslag

Vi starter med å realisere (de ikke-svake) entitetene:

```
bygg(byggnr, navn, nummer, gate, postnr)
aktivitet(navn, tidspunkt)
```

Deretter realiserer vi den svake entiteten ROM, hvor vi dropper å ta med den utledbare attributten Etasje og attributten Utsyr (da denne er multivalued):

```
rom(byggnr, romnr)
```

hvor `rom(byggnr)` refererer til `bygg(byggnr)`, og fanger den identifiserende relasjonen INNEHOLDER. Vi må nå realisere relasjonene mellom entitene. Vi starter med RESEPSJON. Dette er en en-til-en relasjon, så vi har tre valg. Det mest naturlige er kanskje å putte relasjonen inn i tabellen til entiteten BYGG, ettersom det er naturlig at de fleste bygg har en resepsjon. Vi oppdaterer da `bygg`-relasjonen vår til å inneholde primærnøkkelen til ROM, samt attributten Åpningstid som er knyttet til relasjonen. Vi får da oppdatert `bygg` til følgende:

```
bygg(byggnr, navn, nummer, gate, postnr, rBygg, rRom, rÅpningstid)
```

hvor `bygg(rBygg, rRom)` refererer til `rom(byggnr, romnr)`[1].

Deretter vil vi realisere den siste relasjonen STED, som er en mange-til-mange-relasjon, og må derfor få sin egen relasjon:

`sted(aktivitet, byggnr, romnr)`

hvor `sted(aktivitet)` refererer til `aktivitet(navn)` og `sted(byggnr, romnr)` refererer til `rom(byggnr, romnr)`.

Til slutt må vi realisere den multi-valued attributten Utstyr, som da får sin egen relasjon:

`utstyr(byggnr, romnr, utstyr)`

hvor `utstyr(byggnr, romnr)` refererer til `rom(byggnr, romnr)`.

Hele skjemaet ser da slik ut:

```
bygg(byggnr, navn, nummer, gate, postnr, rBygg, rRom, rÅpningstid)
rom(byggnr, romnr)
aktivitet(navn, tidspunkt)
sted(aktivitet, byggnr, romnr)
utstyr(byggnr, romnr, utstyr)
```

hvor

`bygg(rBygg, rRom)` refererer til `rom(byggnr, romnr)`,
`rom(byggnr)` refererer til `bygg(byggnr)`,
`sted(aktivitet)` refererer til `aktivitet(navn)`,
`sted(byggnr, romnr)` refererer til `rom(byggnr, romnr)`, og
`utstyr(byggnr, romnr)` refererer til `rom(byggnr, romnr)`.

SQL (50)

I disse oppgavene skal vi bruke følgende databaseskjema som beskriver spiselige produkter og hva som kreves for å lage produktene (primærnøklene er understreket):

```
produkt(produktID, navn, tid)
ingrediens(ingrediensID, navn, pris)
maskin(maskinID, navn, plassering)
ingrediens_for(ingrediensID, produktID, antall)
brukes_i(maskinID, produktID)
```

med følgende fremmednøkler:

[1] Merk at nå vil alltid `bygg`s attributter `byggnr` og `rBygg` ha lik verdi for alle kolonner, siden resepsjonen til et bygg er et rom i det samme bygget. Dette er jo litt overflødig, men i ER kan vi ikke utrykke at dette er overflødig. Slike overflødige kolonner kan derimot heller fjernes i etterkant av realiseringen.

`ingrediens_for(ingrediensID)` refererer til `ingrediens(ingrediensID)`
`ingrediens_for(produktID)` refererer til `produkt(produktID)`
`brukes_i(maskinID)` refererer til `maskin(maskinID)`
`brukes_i(produktID)` refererer til `produkt(produktID)`

Et produkt består av en unik `produktID` og et navn gitt ved en streng, og et heltall større enn 0 som angir antall sekunder prosessen for å lage produktet tar; en ingrediens består av en unik `ingrediensID`, et navn gitt ved en streng, og en pris per stykk gitt ved en positiv float; maskin har en unik ID gitt ved `maskinID`, et navn gitt ved en streng, samt en unik plassering (gitt ved en kode som i databasen representeres som en streng med maksslengde 7) i fabrikken; for å lage et produkt brukes en mengde ingredienser, og disse er beskrevet i relasjonen `ingrediens_for`, merk at en ingrediens kan bli brukt flere ganger i samme produkt beskrevet av attributten `antall` (hvor det er snakk om et antall i henhold til en passende oppmåling, f.eks. kan det for vann være snakk om milliliter, for mel kan det være gram, mens for gjær kan det være pakker); i tillegg til ingredienser kreves ulike maskiner for å lage et produkt, og hvilket maskin som kreves av hvilket produkt er beskrevet i relasjonen `brukes_i`.

For eksempel kan en database over skjema over se slik ut:

`produkt`

produktID	navn	tid
0	Ris	480
1	Brød	10800
2	Knekk	600
3	Marsipan	1200

`ingrediens`

ingrediensID	navn	pris
0	Riskorn	5.5
1	Vann	1.0
2	Mandel	9.9
3	Sukker	3.0
4	Mel	1.9
5	Gjær	1.2
6	Salt	1.8

`maskin`

maskinID	navn	plassering
0	Kjele	L-324-N
1	Panne	L-110-S
2	Elter	L-956-S
3	Saltkvern	L-998-V
4	Kaffekvern	L-778-Ø
5	Ovn	L-324-S

ingrediens_for

ingrediensID	produktID	antall
0	0	8
1	1	10
1	0	8
2	3	15
3	2	20
3	3	15
4	1	10
5	1	1
6	1	2
6	0	1

brukes_i

maskinID	produktID
0	0
1	2
2	1
2	3
3	1
5	1

Oppgave 3 (10)

Skriv et SQL-script som oppretter skjemaet beskrevet over. Alle skrankene beskrevet i teksten over skal med i skjemaet.

Løsningsforslag

```
CREATE TABLE produkt(
    produktID int PRIMARY KEY,
    navn text,
    tid int CHECK (tid > 0)
);
CREATE TABLE ingrediens(
    ingrdiensID int PRIMARY KEY,
    navn text,
```

```
    pris float CHECK (pris > 0)
);
CREATE TABLE maskin(
    maskinID int PRIMARY KEY,
    navn text,
    plassering varchar(7) UNIQUE
);
CREATE TABLE ingrediens_for(
    ingrediensID int REFERENCES ingrediens(ingrediensID),
    produktID int REFERENCES produkt(produktID),
    antall int,
    PRIMARY KEY (ingrediensID, produktID)
);
CREATE TABLE brukes_i(
    maskinID int REFERENCES maskin(maskinID),
    produktID int REFERENCES produkt(produktID),
    PRIMARY KEY (maskinID, produktID)
);
```

Oppgave 4 (5)

Skriv en spørring som finner alle produkter som lages av en prosess som enten bruker mindre enn 60 eller mer enn 120 sekunder. Skriv opp navnet på produktet og tiden.

Løsningsforslag

```
SELECT navn, tid
FROM produkt
WHERE tid <= 60 OR tid >= 120;
```

Oppgave 5 (5)

Skriv en spørring som finner navnet på alle produkter som har minst én inngrediens, men som ikke trenger noen maskiner.

Løsningsforslag

```
SELECT navn
FROM produkt
WHERE produktID IN (
    SELECT if.produktID FROM ingrediens_for AS if
    EXCEPT
```

```
    SELECT bi.produktID FROM brukes_i AS bi
);
```

eller

```
SELECT DISTINCT p.navn
FROM produkt AS p
     INNER JOIN ingrediens_for AS if USING (produktID)
WHERE produktID NOT IN (
    SELECT bi.produktID FROM brukes_i AS bi
);
```

Oppgave 6 (5)

Skriv en spørring som finner antallet produkter som inneholder ingrediensen med navnet `'salt'` og krever en maskin med `'kvern'` som del av navnet.

Løsningsforslag

```
SELECT count(*) AS antall
FROM produkter AS p
     INNER JOIN ingrediens_for AS if USING (produktID)
     INNER JOIN ingrediens AS i USING (ingrediensID)
     INNER JOIN brukes_i AS bi USING (produktID)
     INNER JOIN maskin AS m USING (maskinID)
WHERE i.navn = 'salt' AND m.navn LIKE '%kvern%';
```

Oppgave 7 (5)

Skriv en SQL-kommando som setter inn et nytt tuppel `(m, p)` i `brukes_i` slik at `m` er `maskinID`en til maskinen med navn `'Saltkvern'` og `p` er produktet med navn `'Havsalt'`.

Løsningsforslag

```
INSERT INTO brukes_i
SELECT m.maskinID, p.produktID
FROM maskin AS m, produkt AS p
WHERE m.navn = 'Saltkvern' AND p.navn = 'Havsalt';
```

Oppgave 8 (10)

Materialkostnaden til et produkt antar vi er lik prisen til alle ingrediensene som inngår i produktet. Skriv en SQL-kommando som lager et view som inneholder materialkostnaden til hvert produkt. Viewet skal inneholde `produktID`en,

navnet, materialkostnaden for hvert produkt, samt antall ingredienser som inngår i produktet. Gi viewet navnet `materialkostnad` og kolonnene henholdsvis `produktID`, `navn`, `kostnad` og `antall`. Sorter viewet på materialkostnad fra dyrest til billigst.

Løsningsforslag

```
CREATE VIEW
  materialkostnad(produktID, navn, kostnad, antall)
AS
SELECT p.produktID, p.navn,
       sum(i.pris * if.antall) AS kostnad,
       count(*) AS antall
FROM produkt AS p
     INNER JOIN ingrediens_for AS if USING (produktID)
     INNER JOIN ingrediens AS i USING (ingrediensID)
GROUP BY p.produktID, p.navn -- p.navn trengs ikke her
ORDER BY kostnad DESC;
```

Oppgave 9 (10)

Total kostnad for å produsere hvert produkt er materialkostnaden pluss 100 kroner per maskin som kreves i produksjonen av produktet pluss 10 kroner per sekund det tar å lage produktet (fra `tid`-kolonnen i `produkt`). Skriv en spørring som finner `produktID`en på de 5 dyreste produktene å produsere. Skriv ut `produktID`, navn og total kostnad per produkt. Du kan bruke viewet `mateialkostnad(produktID, navn, kostnad, antall)` som ble laget i forrige oppgave. Du kan anta at alle produkter bruker minst én maskin og har minst én ingrediens.

Hint: Det kan være lurt å finne kostnaden for maskinene og kostnaden for tiden i hver sin delspørring først, og så til slutt plusse alt dette sammen med materialkostnaden.

Løsningsforslag

```
WITH
  tidskostnad AS (
    SELECT produktID, 10*tid AS kostnad FROM produkt
  ),
  maskinkostnad AS (
    SELECT produktID, 100 * count(*) AS kostnad
    FROM brukes_i
    GROUP BY produktID
```

```
  )
SELECT mk.produktID,
       p.navn,
       (mk.kostnad + t.kostnad + mak.kostnad) AS kostnad
FROM materialkostnad AS mk
     INNER JOIN maskinkostnad AS mak USING (produktID)
     INNER JOIN tidskostnad AS t USING (produktID)
     INNER JOIN produkt AS p USING (produktID)
ORDER BY kostnad DESC
LIMIT 5;
```

Relasjonsmodellen (20)

Oppgave 10 (5)

Gitt følgende relasjon:

$$P(A, B, C, D, E, F, G)$$

og følgende FDer:

$$\begin{aligned} C &\to A, E \\ A, B &\to D \\ B &\to C \\ G &\to B \end{aligned}$$

Hvilke kandidatnøkler har P? Vis hvordan du kommer frem til svaret.

Løsningsforslag

Vi starter med å finne alle attributter som aldri forekommer på noen høyreside, og dermed må være med i alle kandidatnøkler. Av disse har vi FG. Vi finner så alle attributter som kun forekommer på høyresider, ettersom disse aldri kan være med i noen kandidatnøkkel. Av disse har vi DE.

Vi starter så med å sjekke om FG er en kandidatnøkkel: $FG^+ = FGBCAED = ABCDEFG$ som jo er alle attributter. Altså er FG en kandidatnøkkel. Siden enhver annen kandidatnøkkel måtte utvidet FG, ville den ikke vært minimal, så FG må være den eneste kandidatnøkkelen.

Oppgave 11 (5)

Gitt følgende relasjon:

$$R(A, B, C, D)$$

og FDene:

$$A \to B$$
$$B \to C$$

Hva er normalformen til R? Forklar hvordan du kommer frem til svaret.

Løsningsforslag

Vi starter med å finne kandidatnøklene til R: A og D er de eneste attributten som aldri forekommer på en høyreside, så disse må være med i alle kandidatnøkler. C er den eneste som kun forekommer på en høyreside, så kan ikke være med i noen kandidatnøkler. Prøver først med AD. $AD^+ = ADBC$. AD er altså en kandidatnøkkel, og dermed den eneste.

Vi går så gjennom de ulike FDene, og bruker algoritmen for å sjekke Rs normalform:

1. $A \to B$: A er ikke en supernøkkel, så FDen bryter med BCNF. B er ikke en nøkkelattributt, så FDen bryter også med 3NF. A er en del av en kandidatnøkkel, så FDen bryter også med 2NF, og vi ender opp med 1NF. (Kan egentlig stoppe her siden vi nå vet at R er på 1NF, men viser den neste også for ordens skyld.)

2. $B \to C$: B er ikke en supernøkkel, så bryter med BCNF. C er ikke en nøkkelattributt, så FDen bryter også med 3NF. B er ikke en del av en kandidatnøkkel, så vi ender på 2NF.

Den laveste normalformen vi fikk for gjennomgangen var 1NF, altså er R på 1NF.

Oppgave 12 (10)

Gitt relasjonen:

$$T(A, B, C, D)$$

med følgende FDer:

$$A \to B$$
$$B \to C$$
$$C \to D$$

Dekomponer relasjonen T tapsfritt til BCNF ved å bruke algoritmen for tapsfri dekomposisjon.

Løsningsforslag

Vi starter med å finne kandidatnøklene til T: A er den eneste attributten som forekommer kun på venstresider, så den må være med i alle kandidatnøkler. Videre er D den eneste som kun forekommer på høyresidene, så den kan ikke være med i noen kandidatnøkler. Vi starter med å forsøke $A^+ = ABCD$, som er alle attributter, ergo er A en kanidatnøkkel. Ettersom den er den eneste som må forekomme i alle nøkler er A minimal, og dermed den eneste kandidat nøklen. Starter dekomposisjonen med en vilkårlig FD, f.eks. $A \to B$:

1. $A \to B$: Vi vet at A er en supernøkkel, så denne bryter ikke med BCNF. Vi kan dermed fortsette.

2. $B \to C$: B er ikke en supernøkkel, så FDen bryter med BCNF. Vi finner så $B^+ = BCD$, så vi får da to nye tabeller:

$$S_1(B^+) = S_1(B, C, D)$$
$$S_2(B, (ABCD) \setminus B^+)) = S_2(B, A)$$

   For S_1 gjelder følgende FDer:

$$B \to C$$
$$C \to D$$

   mens for S_2 gjelder kun $A \to B$ som ikke bryter med BCNF (ettersom A er kandidatnøkkel også for S_2), og er dermed på BCNF. Det gjenstår å dekomponere S_1 videre.

3. For S_1 får vi (ved samme metode som for T) B som eneste kandidatnøkkel. Sjekker så de gjenstående FDene, starter med f.eks. $B \to C$: B er jo en supernøkkel for S_1, så denne bryter ikke med BCNF. Fortsetter med $C \to D$: C er ikke en supernøkkel, altså brudd på BCNF, må derfor dekomponere. Finner først $C^+ = CD$. Får da følgende dekomposisjon:

$$S_{11}(C^+) = S_{11}(C, D)$$
$$S_{12}(C, (BCD) \setminus C^+) = S_{12}(C, B)$$

   For S_{11} holder kun FDen $C \to D$, men siden C er en kandidatnøkkel (enkelt å se ved samme metode som over) vil denne FDen ikke bryte BCNF. Akkurat det samme kan sier om S_{12}. Dekomposisjonen er altså ferdig.

Resultatet er altså:

$$S_2(B, A)$$
$$S_{11}(C, D)$$
$$S_{12}(C, B)$$