Eksamen IN2090H20

UNIVERSITET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i IN2090

Dato: 16 desember 2020

Tid: 09:00–13:00

Sted: Hjemmeeksamen

Tillatte hjelpemidler: Alle hjelpemidler tillatt

Det er viktig at du leser denne forsiden nøye før du starter.

Generelt:

- Det er viktig at du sjekker emnets semesterside jevnlig. Eventuelle viktige beskjeder under eksamen blir gitt direkte fra faglærer på semestersiden.
- Husk at besvarelsen skal være anonym, du skal ikke oppgi navnet ditt.
- Alle hjelpemidler er tillatt ved skriftlig hjemmeeksamen. Det er opp til deg å innhente informasjon fra tilgjengelige kilder, vurdere kvaliteten og sette det hele sammen til en besvarelse basert på egen bearbeiding av stoffet. Besvarelsen som leveres skal reflektere eget kunnskapsnivå.
- Det er ikke tillatt å sammarbeide eller snakke med andre under eksamen, hverken fysisk eller virtuelt (Zoom, chat, el.). Det eneste unntaket fra dette er med faglærer eller andre fra administrasjonen på IFI (f.eks. i trøsterunden, eller om du har teksninske problemet, el.). Man kan trekkes ut til samtale for å kontrollere eierskap til sin besvarelse https://www.mn.uio.no/om/hms/koronavirus/kontrollsamtale/. Samtalen har ikke innvirkning på sensuren/karakteren, men kan lede til at instituttet oppretter fuskesak. Les mer om hva som regnes som fusk på UiOs nettsider: https://www.uio.no/om/regelverk/studier/studier-eksamener/fuskesaker/
- For øvrig gjelder informasjonen på nettsiden om eksamensavvikling ved MN høsten 2020: https://www.mn.uio.no/om/hms/koronavirus/eksamen-2020.html

Modellering vil enten foregå med et egnet program på egen laptop (f.eks. Dia), eller via håndtegning på papir. Bruk dine foretrukne verktøy til dette, f.eks. bilde av papirark fra mobil. Send bildet fra mobilen til deg selv på mail. Åpne bildet fra mailen på din "eksamensmaskin" og lagre bildet på denne maskinen. Herfra kan du laste opp fila til eksamensoppgaven. Det er viktig at tegningene er leselige. Husk å spesifisere hvilke tegninger som tilhører hvilke oppgaver. Det er studentens ansvar å dobbeltsjekke at bildefil er lastet opp og er leselig. MN anbefaler at du tar vare på originaltegninger til sensur har falt. Du kan også bruke nettbrett/elektronisk penn til å levere håndtegning. Det vil bli gitt 30 min. ekstratid på eksamen som følge av dette.

Trøsterunde (hvor man kan stille spørsmål underveis i eksamen) vil foregå på Zoom, og starte én time etter at eksamen har startet. Dersom dere har spørsmål vil dere da gå inn på Zoom-møtet (lenke vil bli publisert på semestersiden før eksamen), og bli satt i venterom. Deretter vil jeg ta inn én og én etter tur. Ettersom det er mange studenter i dette kurset er det viktig at man er forberedt og har klar alle spørsmål når dere kommer inn i venterommet. Dere kan naturligvis fortsette å jobbe med oppgavene mens dere er i venterommet.

Eksamen består av 3 deler (maksimal poengsum i parentes):

- Modellering (40)
- SQL (40)
- Normalformer og dekomposisjon (20)

På flervalgsoppgavene i oppgave 5 og 11, er det minuspoeng for galt svar. Men den totale poengsummen for hver oppgaver blir aldri lavere enn 0.

Modellering (40%)

Lag en ER- (eller ORM2-) modell for en database for en nettbutikk.

- Du må gjerne dele modellen opp i deler (f.eks. over flere sider), men sørg for at det fremkommer tydelig hvilke deler av modellen som besvarer hvilke oppgaver.
- Du må gjerne inkludere kommentarer i modellene. Dersom det er uklarheter eller tvetydigheter i oppgavebeskrivelsen, bruk sunn fornuft og skriv en kommentar til modellene dine om hvilke antagelser og tolkninger du eventuelt gjør.
- Dersom du ser at det er mulig å modellere deler av oppgaven på flere ulike måter, diskuter kort hvordan din fremgangsmåte er i forhold til de ulike alternativene.
- Tegn og skriv tydelig.

1 Brukere (5)

Lag en modell som fanger grunnleggende informasjon om brukerne av nettbutikken:

For hver bruker skal databasen kunne lagre brukernavn til hver bruker (brukernavnet er unikt blant brukere), navnet til brukeren, og email-adressen brukeren benyttet for å lage kontoen (email-adressen er også unik blant brukere). Videre skal databasen kunne lagre informasjon om kredittkort som brukerne kan benytte for kjøp i nettbutikken. Databasen skal kunne lagre høyst ett kredittkort per bruker, og et kredittkort skal ikke kunne benyttes av fler brukere (altså, en bruker kan ha ingen eller ett kredittkort og et kredittkort kan bli brukt av ingen eller én bruker). Databasen skal lagre datoen brukeren registrerte kredittkortet i nettbutikken. Videre skal kredittkortnummeret og utløpsdatoen for kortet lagres.

Løsningsforslag

Se figur 1 eller figur 2 (ORM2).

2 Produkter (15)

Utvid modellen med informasjon om produkter solgt på nettbutikken, og produktene kjøpt av brukerne:

For produkter solgt i nettbutikken skal databasen lagre navnet til produktet (f.eks. "Samsung TV 65 inch") og en unik identifikator til produktet (f.eks. p1283). Produkter er solgt av en leverandør (en leverandør kan selge mange produkter, og samme produkt kan bli solgt av ulike leverandører). For hver leverandør skal databasen lagre en unik ID (f.eks. s112) og en mengde epost-adresser som skal brukes for kontakt med leverandøren (f.eks. en salgs-email-adresse, en kunde-kontakt-email-adresse, osv.).

Brukere kjøper produkter ved å legge inn bestillinger i nettbutikken. En bruker kan legge inn flere bestillinger. En bestilling kan derimot kun bli lagt inn av én bruker. Ingen bestillinger skal kunne lagres i databasen uten å ha blitt plassert av en bruker. For hver bestilling skal følgende informasjon lagres: bestillingsnummeret (unikt), datoen bestillingen ble lagt inn, prisen på bestillingen, adressen bestillingen skal leveres til (inneholder land, by, postkode, og gateadresse som igjen består av gatenavn og nummer). Videre består en bestilling av en eller fler bestillingslinjer (dette er mekanismen som tillater en bestilling å inneholde fler produkter).

Hver bestillingslinje inneholder nøyaktig ett produkt, nøyaktig én leverandør (leverandøren som selger produktet), og antall enheter av det produktet som er bestilt. Hver bestillingslinje har et nummer som er unikt innen samme bestilling. For eksempel, en bestilling med id 0123 inneholder to bestillingslinjer: bestillingslinje #1 består av 3 enheter av produktet p1283 solgt av leverandør s112; bestillingslinje #2 består av 1 enhet av produkt p3245 solgt av leverandør s122.

Løsningsforslag

Se figur 1 eller figur 2 (ORM2).

3 Ønskelister, anbefalinger og anmeldelser (10)

Utvid modellen videre med informasjon om produkt-ønskelister, anbefalinger og anmeldelser på nettbutikken:

Databasen skal kunne lagre:

- Produktanmeldelser: En bruker kan anmelde flere produkter og det samme produktet kan bli anmeldt av mange brukere. For hver anmeldelse skal

databasen lagre antall stjerner brukeren ga det anmeldte produktet, sammen med en tekstuell beskrivelse.

- Ønskelister: En bruker skal kunne lage mange ønskelister, men en ønskeliste kan kun tilhøre én bruker. Hver ønskeliste skal ha et navn som er unikt blant brukerens ønskelister. I tillegg skal det lagres datoen brukeren laget ønskelisten. En ønskeliste kan inneholde mange produkter. For eksempel, brukeren "joedoe" lager en ønskeliste med navn "Bøker" som inneholder produktene Bok1, Bok2 og Bok3. Naturligvis skal andre brukere kunne lage ønskelister som inneholder de samme produktene som ønskelisten til "joedoe".
- Produktanbefalinger: En bruker skal kunne anbefale produkter til andre brukere. En bruker kan anbefale mange produkter til samme bruker, et produkt kan anbefales av mange brukere til samme bruker, og en bruker kan bli anbefalt mange produkter av en bruker (hint: bruk en ternær relasjon). Datoen for anbefalingen skal også lagres.

Løsningsforslag

Se figur 1 (ER) eller figur 2 (ORM2).

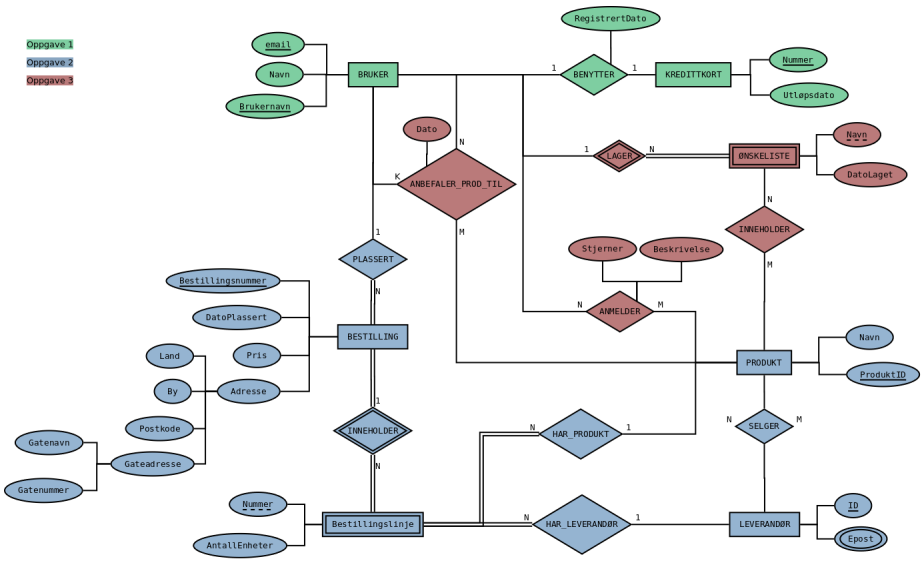


Figure 1: Løsningsforslag modellering ER

4 Realisering (10)

Realiser følgende modell (enten ER-modellen eller ORM2-modellen) til et relasjonelt databaseskjema. Det resulterende databaseskjemaet skal være korrekt (ekvivalent med modellen), effektivt (unngå overflødige tabeller), og tydelig (bør

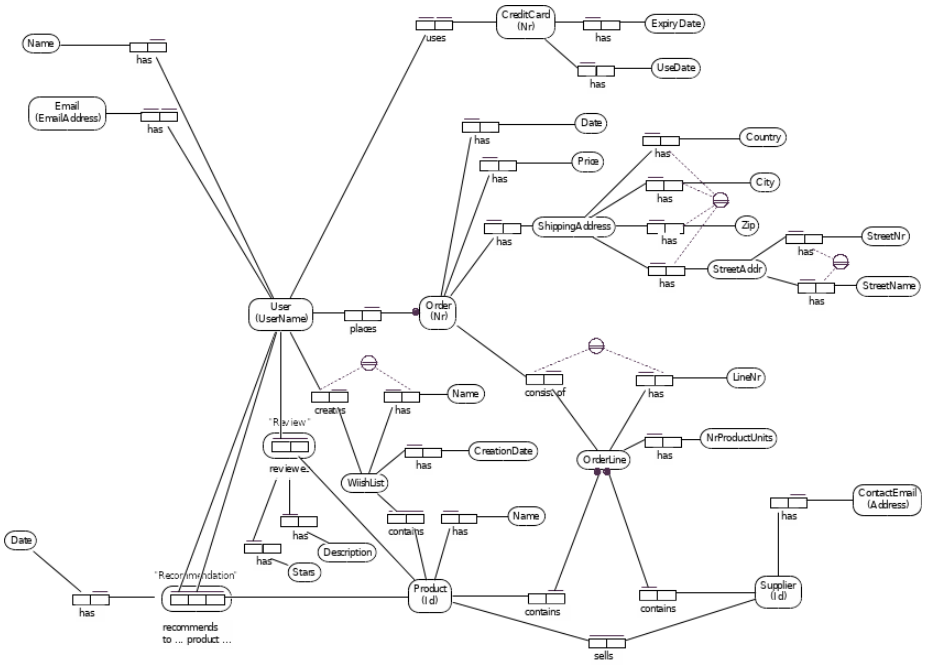
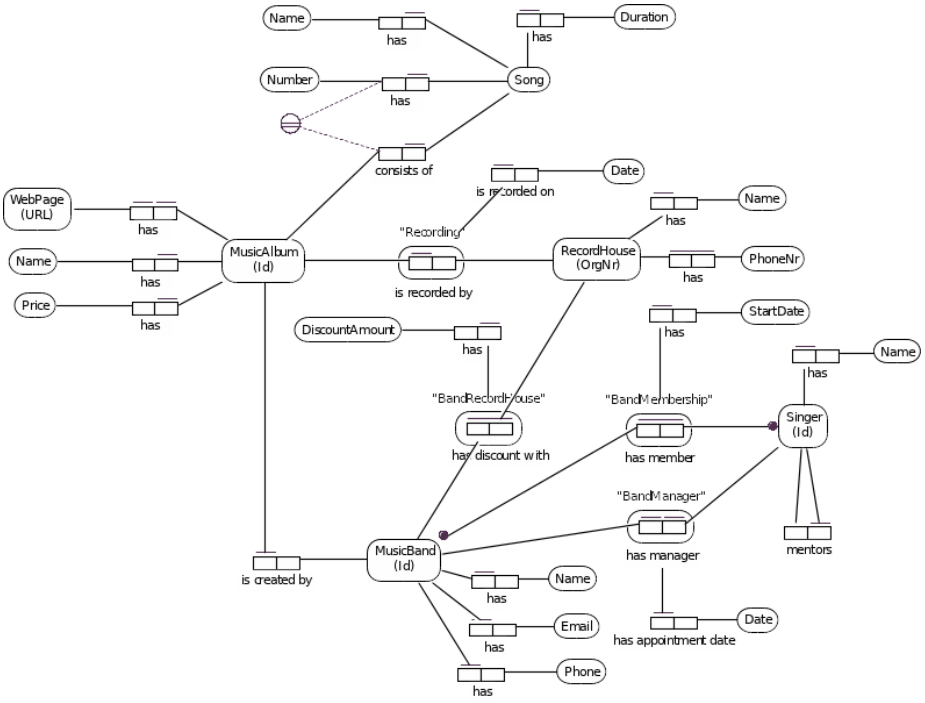


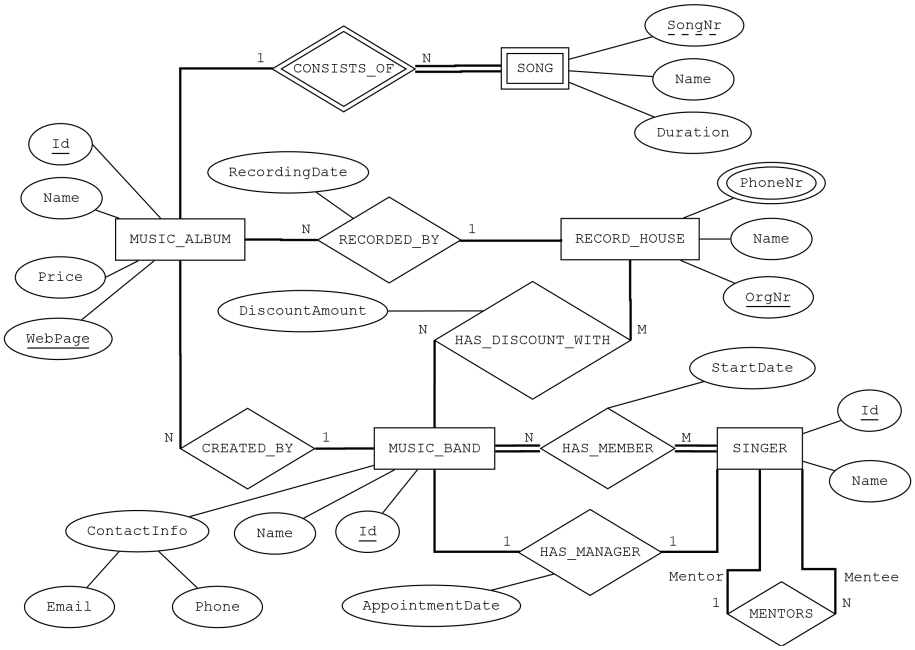
Figure 2: Løsningsforslag modellering ORM2

være lett å forstå). Bruk algoritmen for realisering for å lage et slikt skjema, og forklar eventuelle valg du tar underveis.

For hver relasjon, oppgi relasjonens navn, og navnet til hvert attributt. Du trenger ikke oppgi datatyper for attributtene, og du skal ikke bruke SQL i denne oppgaven. Bruk understreking for å markere kandidatnøkler. Dersom det er flere kandidatnøkler, bruk fet skrift for å markere valgt primærnøkkel. Bruk piler for å markere fremmednøkler (f.eks. "T(A) -> S(B)" for å uttrykke at relasjon T sin attributt A er en fremmednøkkel som peker på relasjon S sin B attributt).

Om du ønsker å ha bilde større eller i en egen tab, høyreklikk på bildet og velg "Se bildet/View Image".





Løsninsforslag

- *MusicAlbum*(**Id**, *Webpage*, *Name*, *Price*, *RecordedBy*, *RecordedDate*, *CreatedBy*)

- $RecordHouse(\underline{OrgNr}, Name)$
- $MusicBand(\underline{Id}, Name, Email, Phone, Manager, AppointmentDate)$
- $Singer(\underline{Id}, Name, Mentor)$
- $Song(\underline{SongNr}, Album, Name, Duration)$
- $HasMember(\underline{Band}, \underline{Singer}, StartDate)$
- $HasDicountWith(\underline{RecordHouse}, \underline{Band}, DiscountAmount)$
- $RecourdHousePhone(\underline{RecordHouse}, \underline{PhoneNr})$

hvor

- $Song(Album) \rightarrow MusicAlbum(Id)$
- $MusicAlbum(RecordedBy) \rightarrow RecordHouse(OrgNr)$
- $Band(Manager) \rightarrow Singer(Id)$
- $Singer(Mentor) \rightarrow Singer(Id)$
- $MusicAlbum(CreatedBy) \rightarrow MusicBand(Id)$
- $HasMember(Band) \rightarrow MusicBand(Id)$
- $HasMember(Singer) \rightarrow Singer(Id)$
- $HasDiscountWith(RecordHouse) \rightarrow RecordHouse(OrgNr)$
- $HasDiscountWith(Band) \rightarrow MusicBand(Id)$
- $RecordHousePhone(RecordHouse) \rightarrow RecordHouse(OrdNr)$

Valg:

- *Id* som primærnøkkel for *MusicAlbum*.
- Realiserer `HAS_MANAGER` som attributt *Manager* i *Band*, siden det virker mer sannsynlig at et band har en manager enn at en bestemt sanger er manager for et band. Dette vil derfor redusere antall `NULL`-verdier i databasen.
- Realiserer `RECORDED_BY` som attributt *RecordedBy* i *MusicAlbum* for å redusere antall relasjoner
- Realiserer `MENTORS` som attributt i *Singer*, og antar at det er vanlig å ha mentorer
- Realiserer `CREATED_BY` som attributt i *MusicAlbum*, og antar at det er vanlig at album er laget av band

SQL (40%)

Oppgavene i denne delen er om SQL. Alle oppgavene vil bruke det samme databaseskjemaet, og en beskrivelse av dette er vedlagt som PDF for hver oppgave.

Du må gjerne inkludere kommentarer i spørringene (gjøres med "–" i SQL). Dersom det er uklarheter eller tvetydigheter i oppgavebeskrivelsen, bruk sunn fornuft og skriv en kommentar om hvilke antagelser og tolkinger du gjør.

1 Flervalg (5)

Dersom dataene i databasen er lik eksempeldataene i PDFen, hvilke av følgende tupler vil være med i resultatet på følgende spørring:

```
SELECT by.navn AS by, poi.navn AS land
FROM by INNER JOIN poi USING (bid)
WHERE by.navn LIKE '%r%i%' OR
      poi.navn LIKE '%er';
```

1. (Berlin, Berlinmuren)
2. (Oslo, Slottet)
3. (Kristiansand, Kilden)
4. (Paris, Eiffeltårnet)
5. (Oslo, Bentsebrua)
6. (London, London Bro)
7. (Berlin, Frihetsgudinnen)
8. (Oslo, Aker Brygge)
9. (Bergen, Lyse Kloster)
10. (Roma, Caracallas Termer)
11. (Firenze, Villa Bardini)
12. (Kristiansand, Maemo)

Løsningsforslag

1, 3, 4, 9, 10

2 Italiensk værmelding (5)

Skriv en SQL-spørring som finner morgendagens (17.12.2020) værmelding for alle byer i Italia. Spørringen skal skrive ut navnet på byen, antall millimeter regn og vindstyrken. Sorter resultatet alfabetisk på bynavn.

Løsningsforslag

```
SELECT b.navn, v.nedbør, v.vind
FROM land AS l
     INNER JOIN by AS b USING (lid)
     INNER JOIN værmelding AS v USING (bid)
WHERE l.navn = 'Italia' AND v.dato = '2020-12-17'
ORDER BY b.navn;
```

3 Ukesmelding for jula (5)

Skriv en SQL-spørring som finner navn, totalt antall millimeter nedbør og gjennomsnittlig vindstyrke i romjulsuka (altså fra og med dato 24.12.2020 til og med dato 31.12.2020) for hver by.

Løsningsforslag

```
SELECT b.navn, sum(v.nedbør) AS sum_nedbør, avg(v.vind) AS avg_vind
FROM by AS b INNER JOIN værmelding AS v USING (bid)
WHERE v.dato >= '2020-12-24' AND v.dato <= '2020-12-31'
GROUP BY b.bid, b.navn;
```

4 Byer uten regn (5)

Skriv en SQL-spørring som finner navn på alle byer hvor det ikke er meldt noe regn og ikke noe vind fra og med julaften (24.12.2020) og ut året. Spørringen skal skrive ut navnet på byene. Du kan anta at vi har værmelding (og både nedbør og vindstyrke) for alle byer hver dag i julen.

Løsningsforslag

```
SELECT navn
FROM by
WHERE bid NOT IN (
  SELECT bid
  FROM værmelding
  WHERE dato >= '2020-12-24' AND
        dato < '2021-01-01' AND
        (nedbør > 0 OR vind > 0)
);
```

eller

```
SELECT b.navn
FROM by AS b INNER JOIN værmelding AS v USING (bid)
WHERE dato >= '2020-12-24' AND
      dato < '2021-01-01'
GROUP BY b.bid, b.navn
HAVING sum(v.nedbør) = 0 AND sum(v.vind) = 0;
```

5 Steder og vær (10)

Skriv en SQL-kommando som lager et `VIEW` med navn `Steder` som viser dagens værmelding (nedbør i mm. og vindstyrke) for både byer og POIs i samme tabell.

Du kan anta at dagens dato finnes i variabelen `current_date` (slik som i PostgreSQL). VIEWet skal ha 4 kolonner, en med navnet på stedet, en med plassering som er landet dersom stedet er en by og adressen dersom stedet er en POI, samt nedbør og vindstyrke. For POIs er nedbør og vindstyrke lik byen den befinner seg i sin nedbør og vindstyrke. Vi er kun interessert i steder som faktisk har en posisjon, altså skal posisjon aldri være `NULL`. F.eks. kan innholdet i `VIEW`et se slik ut:

```
          navn          |          posisjon          | nedbør | vind
```

```
--------------------------+--------------------------------+--------+------
 Oslo                     | Norge                          |    1.1 |    2
 Kilden                   | Sjølystveien 2                 |    3.4 |    5
 Eiffeltårnet             | 5 avenue Anatole               |      0 |    2
 Grefsenkolloen Restaurant | Grefsenkollveien 100          |    1.1 |    2
 Villa Bardini            | Costa S. Giorgio, 2-4          |      0 |    1
 Berlin                   | Tyskland                       |    0.1 |    0
 Kristiansand             | Norge                          |    3.4 |    5
 Aker Brygge              | Støperigata 2                  |    1.1 |    2
 Roma                     | Italia                         |      0 |    1
 Kaf Kafe                 | Jacobsfjorden 4                |   10.7 |    8
 Feffo                    | Via Delle Fornaci 2/4/6        |      0 |    1
 Maemo                    | Dronning Eufemias gate 23      |    1.1 |    2
 Firenze                  | Italia                         |    1.3 |    6
 Bergen                   | Norge                          |   10.7 |    8
 Lyse Kloster             | Edne 121                       |   10.7 |    8
 Paris                    | Frankrike                      |      0 |    2
 Caracallas Termer        | Viale delle Terme di Caracalla |      0 |    1
 Hamburg                  | Tyskland                       |    3.1 |    0
 Slottet                  | Karl Johansgate 1              |    1.1 |    2
```

Løsningsforslag

```sql
CREATE VIEW steder AS
SELECT b.navn, l.navn AS posisjon, v.nedbør, v.vind
FROM land AS l INNER JOIN by AS b USING (lid)
     INNER JOIN værmelding AS v USING (bid)
WHERE v.dato = current_date -- l.navn er markert med NOT NULL
UNION ALL -- også greit med UNION
SELECT p.navn, p.adresse AS posisjon, v.nedbør, v.vind
FROM poi AS p INNER JOIN by AS b USING (bid)
     INNER JOIN værmelding AS v USING (bid)
WHERE v.dato = current_date AND p.adresse IS NOT NULL;
```

6 Ferieplanlegging (10)

La oss si at du skal reise på ferie til Frankrike og ønsker å finne ut hvilken by du skal reise til for å gå tur og se på museer. Skriv derfor en SQL-spørring som finner den byen i Frankrike med opphold (0 mm. nedbør) i morgen (17.12.2020), som har minst 3 kaféer og som har flest museer.

Løsningsforslag

```sql
WITH
  opphold_og_kafeer AS (
    SELECT b.bid, b.navn
```

```
    FROM land AS l
         INNER JOIN by AS b USING (lid)
         INNER JOIN værmelding AS v USING (bid)
         INNER JOIN poi AS p USING (bid)
    WHERE l.navn = 'Frankrike' AND
          v.dato = '2020-12-17' AND
          v.nedbør = 0 AND
          p.type = 'Kafé'
    GROUP BY b.bid, b.navn
    HAVING count(*) >= 3
  )
SELECT o.navn, count(*) AS antall_museer
FROM opphold_og_kafeer AS o INNER JOIN poi AS p USING (bid)
WHERE p.type = 'Museum'
GROUP BY o.bid, o.navn
ORDER BY antall_museer DESC
LIMIT 1;
```

Alternativ løsning, hvor man antar at "flest museer" gjelder for alle byer i Frankrike (ikke bare dem med opphold og minst 3 kaféer):

```
WITH
  opphold_og_kafeer AS (
    SELECT b.bid, b.navn
    FROM land AS l
         INNER JOIN by AS b USING (lid)
         INNER JOIN værmelding AS v USING (bid)
         INNER JOIN poi AS p USING (bid)
    WHERE l.navn = 'Frankrike' AND
          v.dato = '2020-12-17' AND
          v.nedbør = 0 AND
          p.type = 'Kafé'
    GROUP BY b.bid, b.navn
    HAVING count(*) >= 3
  ),
  antall_museer AS (
    SELECT b.bid, count(*) AS antall_museer
    FROM by AS b
         INNER JOIN land AS l USING (lid)
         INNER JOIN poi AS p USING (bid)
    WHERE p.type = 'Museum' AND l.navn = 'Frankrike'
    GROUP BY b.bid, b.navn
  ),
  flest_museer AS (
    SELECT *
    FROM antall_museer
    WHERE antall_museer = (SELECT max(antall_museer) FROM antall_museer)
```

```
  )
SELECT ok.navn, fm.antall_museer
FROM oppholde_og_kafeer AS ok
     INNER JOIN flest_museer AS fm USING (bid);
```

Relasjonsmodellen og normalformer (20%)

1 Nøkler (5)

Gitt følgende relasjon

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

og følgende FDer:

1. `A -> BC`
2. `BC -> A`
3. `D -> B`
4. `B -> D`

Hvilke kandidatnøkler har relasjonen?

1. `A`
2. `B`
3. `C`
4. `D`
5. `AB`
6. `AC`
7. `AD`
8. `BC`
9. `BD`
10. `CD`
11. `ABC`
12. `ABD`
13. `ACD`
14. `BCD`
15. `ABCD`

Løsningsforslag

`A`, `BC`, `CD`.

2 Normalformer (5)

Gitt følgende relasjon

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

med kandidatnøklene `AB` og `C` og følgende FDer

1. `D -> B`
2. `C -> AD`
3. `B -> D`
4. `A -> C`

Hvilken normalform har relasjonen. Vis hvordan du kommer frem til svaret.

Løsningsforslag

Det er en skrivefeil i oppgaven. A er ihht. FDene en kandidatnøkkel alene, så AB er derfor ikke en kandidatnøkkel.

Dette ble opplyst om under eksamen, men etter at en del hadde løst oppgaven. Det er derfor to mulige løsninger av oppgaven, en hvor man antar at AB og C er kandidatnøklene, og en hvor man antar at A og B er kandidatnøklene.

Løsning om AB og C er kandidatnøkler:

1. `D -> B` (3NF)

- `D` er ikke en supernøkkel, brudd på BCNF.
- `B` er en nøkkelattributt, så ikke brudd på 3NF.

2. (Splitter FDen opp i to):

- `C -> A` (BCNF): `C` er en supernøkkel, ikke brudd på BCNF.
- `C -> D` (BCNF): `C` er en supernøkkel, ikke brudd på BCNF.

3. `B -> D` (1NF)

- `B` er ikke en supernøkkel, brudd på BCNF.
- `D` er en ikke nøkkelattributt, så brudd på 3NF.
- `B` er del av en kandidatnøkkel, så brudd på 2NF.

4. `A -> C` (3NF)

- `A` er ikke en supernøkkel, brudd på BCNF.
- `C` er en nøkkelattributt, så ikke brudd på 3NF.

Altså er `R` på 1NF.

Løsning om A og C er kandidatnøkler:

1. `D -> B` (2NF)

- `D` er ikke en supernøkkel, brudd på BCNF.
- `B` er ikke nøkkelattributt, så brudd på 3NF.
- `D` er ikke en del av en kandidatnøkkel, så ikke brudd på 2NF.

2. (Splitter FDen opp i to):

- `C -> A` (BCNF): `C` er en supernøkkel, ikke brudd på BCNF.
- `C -> D` (BCNF): `C` er en supernøkkel, ikke brudd på BCNF.

3. `B -> D` (2NF)

- `B` er ikke en supernøkkel, brudd på BCNF.
- `D` er en ikke nøkkelattributt, så brudd på 3NF.
- `B` er ikke en del av en kandidatnøkkel, så ikke brudd på 2NF.

4. `A -> C` (BCNF)

- `A` er en supernøkkel, så ikke brudd på BCNF.

Altså er `R` på 2NF.

2 Tapsfri dekomposisjon (10)

Gitt følgende relasjon `Student(id, navn, veileder, oppgave, institutt)`

med kandidatnøkler `{id, oppgave}` og FDer

1. `id -> navn`
2. `oppgave -> veileder`
3. `oppgave -> institutt`

Dekomponer relasjonen tapsfritt til BCNF. Beskriv hvordan du kommer frem til resultatet, og list opp nøkler og FDer for alle tabeller underveis.

Løsningsforslag

Dekomponerer med hensyn på `id -> navn` (`id` ikke supernøkkel så brudd på BCNF) først og får `T1(id, navn)` og `T2(id, veileder, oppgave, institutt)`. For `T1` holder kun den første FDen, og har dermed kandidatnøkkel `id`, og er dermed på BCNF.

For `T2` holder `oppgave -> veileder` og `oppgave -> institutt` og har dermed kandidatnøkkel `{id, oppgave}`. Den første av de to FDene bryter med BCNF. Dekomponerer `T2` videre og får `T21(oppgave, veileder, institutt)` og `T22(id, oppgave)`. For `T21` holder FDene 2. og 3. og har da kandidatnøkkel `oppgave` og er den dermed på BCNF. For `T22` holder ingen av FDene, og den er dermed også på BCNF.

`Student` kan dermed dekomponeres tapsfritt til `T1(id, navn)`, `T21(oppgave, veileder, institutt)` og `T22(id, oppgave)`.