Universitetet i Sørøst-Norge

SENSORVEILEDNING

Utarbeidet av: Bjørn Kristoffersen
Godkjent av: Egil P. Andersen
Dato: 24.06.2022
Versjonsnummer: 2

Tilgjengeliggjøring: Elektronisk

Selve sensorveiledningen legges inn i tekstboksen(e) på s. 2, og det er denne teksten som kommuniseres til studentene.

- En sensorveiledning bør som minimum inneholde en beskrivelse av hva det ut fra emnets læringsmål og innhold forventes at kandidatene må kunne, eller en løsningsskisse/løsningsforslag der hvor dette er hensiktsmessig. Se nedenfor om kravet til tilgjengeliggjøring for eksamensoppgaver som gjenbrukes.

- Sensorveiledningen skal inneholde en beskrivelse av kravet til å bestå eksamen. Veiledningen skal også gi studentene forståelse av hva som skulle til for å få et godt resultat på eksamen. En god sensorveiledning vil dermed kunne bidra til å redusere antallet klager på karakterfastsettelsen.

- Sensorveiledningen skal foreligge samtidig med utarbeidelsen av eksamensoppgaven, og sendes eksamenskontoret senest tre dager før eksamen. Dersom det oppdages forhold under eksamen eller under sensuren som vil påvirke vurderingsgrunnlaget kan det foretas en endring/justering av sensorveiledningen (med angivelse av versjonsnummer).

- Emneansvarlig har ansvaret for å påse at det utarbeides sensorveiledning (kan utarbeides av andre), og skal oversende denne til seksjon for eksamen slik at denne arkiveres sammen med eksamensoppgaven. Justert sensorveiledning skal arkiveres på samme vis.

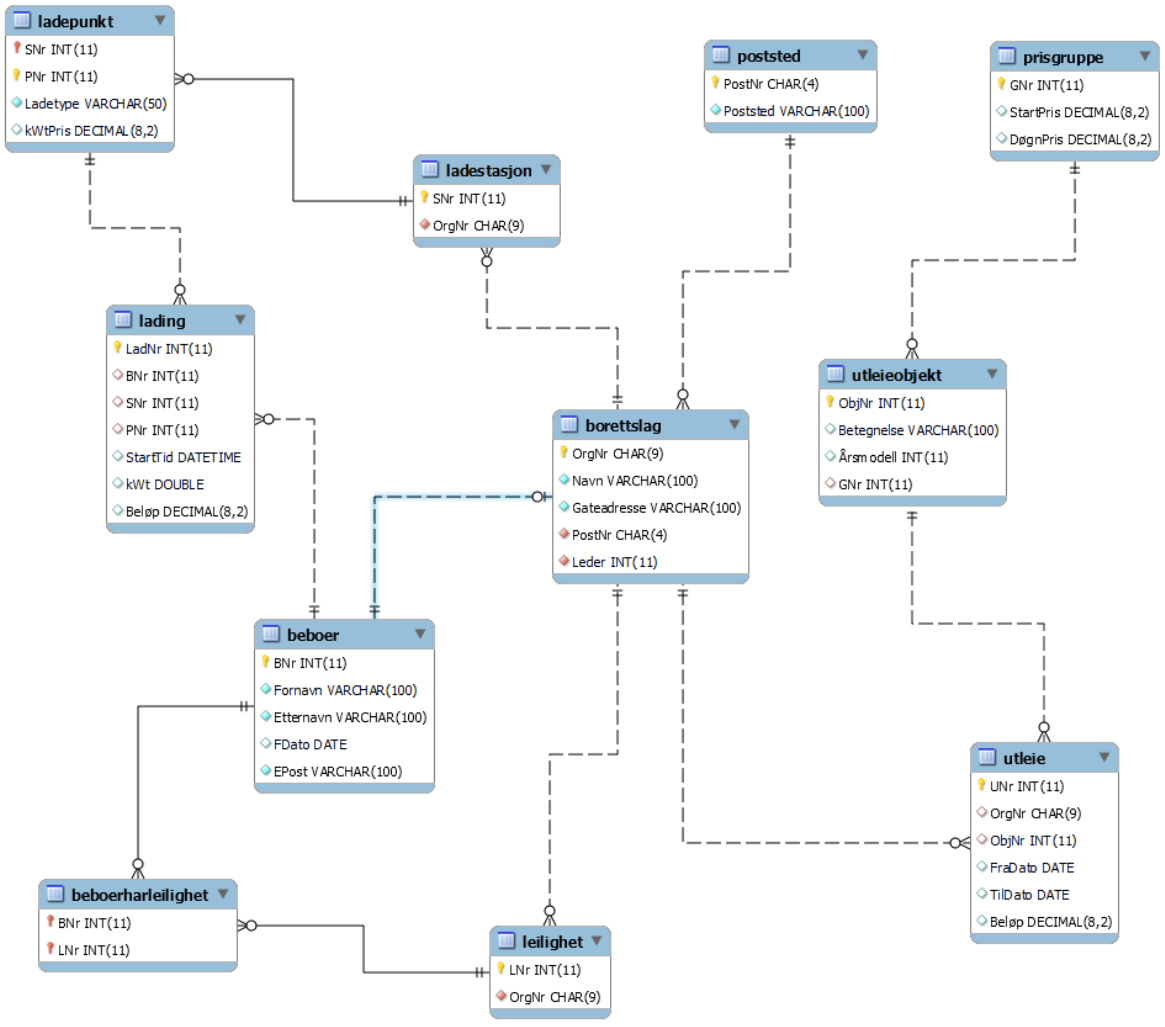
- Sensorveiledninger skal foreligge på undervisningsspråket i emnet.

- Avhengig av type eksamen kan sensorveiledningen inneholde en beskrivelse av sammenhengen mellom oppgaveteksten og læringsutbyttet, hva man har fokusert på i emnet, pensumdekningen, krav til språkføring og referanseteknikk, vektlegging av selvstendighet mm.

Sensorveiledning

Emnekode	Emnenavn	Eksamensdato	Campus	Faglærers navn
DAT1000 / DAT1000N	Database 1	24.05.2022	Bø/Bakkenteigen/Nett	Bjørn Kristoffersen

Oppgave 1



Besvarelser på oppgaver i datamodellering blir vurdert ut fra flere og sammensatte kriterier. Det er for det første snakk om å få med alle momentene som blir nevnt i oppgaveteksten, men man skal på den andre siden heller ikke ta med unødvendige ting. Videre er det viktig at man får samlet de ulike opplysningene (attributtene/kolonnene) med fornuftige datatyper i hensiktsmessige entiteter (tabeller) og får opprettet forhold mellom entiteter der det er hensiktsmessig. Det er samtidig viktig å mestre de «tekniske» sidene ved ER: alle entiteter bør ha en fornuftig primærnøkkel, man bør ta hensiktsmessige valg når det gjelder en-til-mange, en-til-en og mange-til-mange forhold samt når det gjelder identifiserende og ikke-identifiserende forhold. Attributter bør gis naturlige datatyper og være definert med **NOT NULL** der det er aktuelt. Blant annet.

Det er vanskelig å beskrive presist hva som skal til for at en datamodell skal bli vurdert til f.eks. en A eller C eller F, for noen besvarelser kan ha med nesten alle momentene fra oppgaveteksten, men inneholde flere store, tekniske feil, mens andre besvarelser kan være teknisk sett gode, men ikke ha med mange av problemstillingene. Noen ganger kan det være krevende å tolke datamodeller riktig (slik det var ment), fordi ulike personer (studenter) velger forskjellige ord for å beskrive samme ting. Som regel vil det dessuten finnes flere, ulike og fullgode løsninger på en gitt datamodelleringsoppgave.

Gitt alle disse forbeholdene: For å få uttelling tilsvarende karakteren C bør man i hovedsak mestre de tekniske sidene ved ER og også ha med de viktigste momentene. Men man kan altså gjøre noen feil, kanskje til og med en grov feil som f.eks. å «snu» et en-til-mange forhold – hvis man gjort dette riktig andre steder i modellen. For å få karakteren A bør (så å si) alle momentene være med og det bør kun være små tekniske feil. For å få karakteren E kan det være flere grove feil og store mangler, men man må likevel levere et *ER-diagram* som viser entiteter, attributter og forhold for den aktuelle problemstillingen.

Oppgave 2

Det finnes flere gode datamodeller til oppgave 1. Vanskelighetsgraden på SQL-spørsmålene i denne oppgaven vil variere avhengig av løsningen man har valgt i oppgave 1. Dette blir det tatt høyde for under sensur.

2-a

Alle ladepunkter av type hurtiglader med pris pr. kWt mellom 5 og 8 kr. Sortert synkende med hensyn på pris (de dyreste først).

Denne oppgaven tester spørringer mot én tabell, sortering og bruk av AND og BETWEEN.

```
SELECT *
FROM Ladepunkt
WHERE Ladetype = 'Hurtiglader' AND (kWtPris BETWEEN 5 AND 8)
ORDER BY kWtPris DESC;
```

Alternativ løsning:

```
SELECT *
FROM Ladepunkt
WHERE Ladetype = 'Hurtiglader' AND (kWtPris >= 5 AND kWtPris <= 8)
ORDER BY kWtPris DESC;
```

Det er ikke helt opplagt om kr. 8.00 skal være med eller om høyeste pris er 7.99. Begge tolkinger bør godtas.

For de fleste vil denne oppgaven la seg løse med en spørring mot én enkelt tabell. Hvis det er nødvendig å koble tabeller, blir det vurdert om man gjør dette på riktig måte. Lignende helhetsvurderinger blir gjort ved vurdering av alle SQL-spørsmålene.

2-b

Beboere med epostadresse som slutter med gmail.com eller gmail.no. Beboerens navn skal vises i én kolonne med fornavnet forkortet, f.eks. skal Arne Hansen bli vist som A. Hansen. Ta også med beboerens fødselsår i utskriften.

Denne oppgaven tester spørringer mot én tabell og bruk av funksjoner, jokernotasjon (LIKE) og sammensatte betingelser (OR).

```
SELECT BNr, CONCAT(LEFT(Fornavn, 1), '. ', Etternavn) AS Navn, YEAR(FDato) AS Fødselsår
FROM Beboer
WHERE EPost LIKE '%gmail.no' OR EPost LIKE '%gmail.com';
```

2-c

Navn på alle beboere som har en leilighet med postnummer 3200. Sorter utskriften etter navn på borettslag og så etter etternavn innen hvert borettslag. Hvis det er flere med samme etternavn i samme borettslag skal disse sorteres etter fornavn.

Denne oppgaven tester likekoblinger og sortering på flere kriterier.

```
SELECT Bo.Navn, Be.Etternavn, Be.Fornavn
FROM Borettslag AS Bo INNER JOIN
     (Beboer AS Be INNER JOIN
            (Leilighet AS Le INNER JOIN BeboerHarLeilighet AS BL
       ON BL.LNr = Le.LNr)
      ON Be.BNr = BL.BNr)
    ON Le.OrgNr = Bo.OrgNr
WHERE Bo.PostNr = '3200'
ORDER BY Bo.Navn, Be.Etternavn, Be.Fornavn;
```

Spørringen kan alternativt skrives med vanlige WHERE-betingelse. Det gir også full uttelling (her og i andre oppgaver der man må skrive likekoblinger).

```
SELECT Bo.Navn, Be.Etternavn, Be.Fornavn
FROM Borettslag AS Bo, Beboer AS Be, Leilighet AS Le, BeboerHarLeilighet AS BL
WHERE Be.BNr = BL.BNr AND BL.LNr = Le.LNr AND Le.OrgNr = Bo.OrgNr
AND Bo.PostNr = '3200'
ORDER BY Bo.Navn, Be.Etternavn, Be.Fornavn;
```

2-d

Totalt ladebeløp inneværende år for alle beboere. Sortert med hensyn på beboerens navn.

Denne oppgaven tester gruppering og mengdefunksjoner.

Med INNER JOIN notasjon:

```
SELECT B.BNr, B.Fornavn, B.etternavn, SUM(Beløp) AS Totalt
FROM Beboer AS B INNER JOIN Lading AS L ON B.BNr = L.BNr
WHERE YEAR(StartTid) = YEAR(CURDATE())
GROUP BY B.BNr, B.Fornavn, B.etternavn;
```

Med kobling i WHERE:

```
SELECT B.BNr, B.Fornavn, B.etternavn, SUM(Beløp) AS Totalt
FROM Beboer AS B, Lading AS L
WHERE B.BNr = L.BNr
AND YEAR(StartTid) = YEAR(CURDATE())
GROUP BY B.BNr, B.Fornavn, B.etternavn;
```

2-e

Øk ladepris pr. kWt med 10 % for alle hurtigladere.

Denne oppgaven tester bruk av UPDATE.

```
UPDATE Ladepunkt
SET kWtPris = kWtPris * 1.1
WHERE Ladetype = 'Hurtiglader';
```

2-f

Skriv en SQL-oppgave som tester evne til å bruke vekselvirkende delspørringer (correlating subqueries).

Denne oppgaven tester bruk av vekselvirkende delspørringer, samt evne til å se sammenhengen mellom oppgaveformulering og SQL-kode.

Eksempel: Lag en spørring som viser ladepunkter med høyeste kilowatttimepris for sin ladetype.

```
SELECT *
FROM Ladepunkt AS L1
WHERE L1.kWtPris >= ALL
  (SELECT L2.kWtPris FROM Ladepunkt AS L2 WHERE L1.Ladetype = L2.Ladetype);
```

Man kan alternativt bruke MAX i SELECT-delen og droppe ALL.

2-g

Navn på beboere i Løkka borettslag som ikke har benyttet seg av muligheten for lading.

Denne oppgaven tester delspørringer og IN-operatoren.

```
SELECT B.*
FROM Beboer AS B, BeboerHarLeilighet AS BL,
  Leilighet AS L, Borettslag AS Bo
WHERE B.BNr = BL.BNr AND BL.LNr = L.LNr AND L.OrgNr = Bo.OrgNr
AND Bo.Navn = 'Løkka'
AND B.BNr NOT IN
  (SELECT DISTINCT BNr FROM Lading);
```

Her kan man også bruke NOT EXISTS. Og, som ellers, kan man gjerne bruke INNER JOIN notasjon.

2-h

Registrere en ny beboer på en eksisterende leilighet. Velg eksempeldata selv og gjør forutsetninger om eventuelle fremmednøkler.

Denne oppgaven tester bruk av INSERT.

Antar Beboer.BNr er autonummerert og at leilighet 4 er registrert.

```
INSERT INTO
  Beboer(Fornavn, Etternavn, FDato, EPost)
VALUES
  ('Anders', 'Andersen', '1978-23-06', 'andand@gmail.no');

INSERT INTO
  BeboerHarLeilighet(LNr, BNr)
VALUES
  (4, LAST_INSERT_ID());
```

Oppgave 3

3-a

Vi ser altså på følgende tabell:

- Topptur(MedlemsNr, Fornavn, Kjønn, Fjelltopp, Moh, Dato, KlSlett, Temp)

Tabellen inneholder redundans (dobbeltlagring). Dette er generelt uheldig fordi det sløser med lagringsplass og kan ha uheldige konsekvenser når man skal gjøre endringer (oppdateringsanomalier). Eksempler på redundans:

- For gjentatte forekomster av samme medlemsnummer blir fornavn og kjønn gjentatt.
- For gjentatte forekomster av samme fjelltopp blir samme verdi for moh gjentatt.

Hvis man f.eks. finner ut at den målte høyden på en fjelltopp endrer seg (det har skjedd mange ganger opp gjennom historien med en rekke fjelltopper, blant annet på grunn av at man har fått bedre målemetoder), så må Moh på alle radene for denne fjelltoppen oppdateres.

Følgende funksjonelle avhengigheter gjelder, i og med at vi vil tillate at et medlem kan besøke samme fjelltopp flere ganger på samme dag:

- MedlemsNr, Fjelltopp, Dato, KlSlett $\rightarrow$ alle kolonner
- MedlemsNr $\rightarrow$ Fornavn, Kjønn
- Fjelltopp $\rightarrow$ Moh

Kandidatnøkkel: MedlemsNr, Fjelltopp, Dato, KlSlett

Den første avhengigheten er uproblematisk (starter i kandidatnøkkelen).

Avhengigheten MedlemsNr → Fornavn bryter med 2NF. Vi splitter tabellen og får:

- Topptur2(MedlemsNr, Fjelltopp, Moh, Dato, KlSlett, Temp)
- Medlem(MedlemsNr, Fornavn, Kjønn)

Avhengigheten Fjelltopp → Moh bryter også med 2NF. Vi splitter igjen og får sluttresultatet – med primærnøkler og fremmednøkler:

- Måling(MedlemsNr*, Fjelltopp*, Dato, KlSlett, Temp)
- Medlem(MedlemsNr, Fornavn, Kjønn)
- Fjell(Fjelltopp, Moh)

3-b

En god besvarelse bør nevne datatyper, primærnøkler og fremmednøkler, NOT NULL, UNIQUE og CHECK-regler, og gi konkrete eksempler på bruk av teknikkene mot Topptur-tabellen. Det er f.eks. naturlig å sikre at MedlemsNr og Moh er positive heltall, at Dato er en lovlig dato, at Kjønn er en lovlig kode for kjønn, at Temp er innenfor rimelige grenser, osv.

3-c

En god besvarelse bør nevne flere bruksområder for views – bruk i problemløsing for å bryte ned komplekse spørringer i flere «steg», presentasjon av deler av databasen for ulike brukergrupper og bruk i forbindelse med rettigheter/tilgang til deler av databasen – i tillegg til å vise og forklare et nyttig eksempel på view mot egen database fra oppgave 1. For full uttelling bør definisjonen av viewet ha en viss kompleksitet, f.eks. kreve kobling av flere tabeller, gruppering og mengdefunksjoner, delspørringer eller lignende.