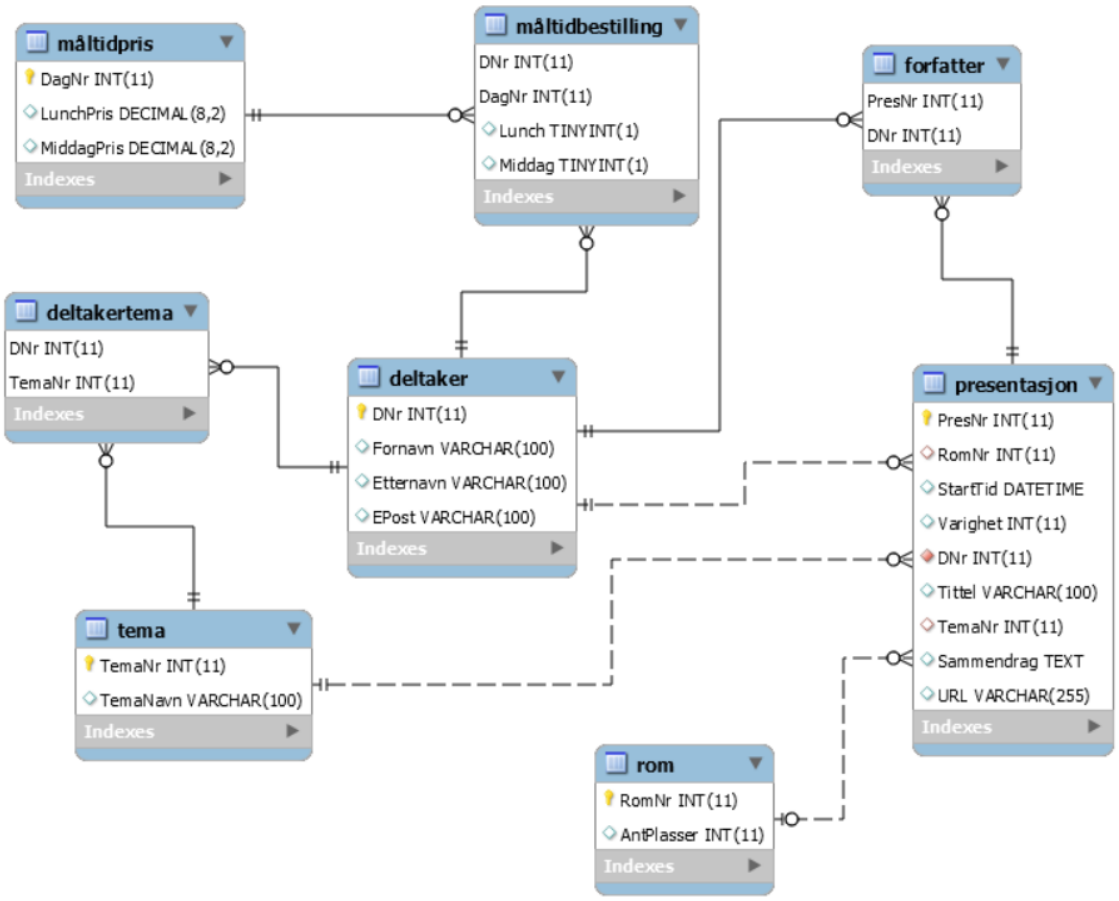
Universitetet i Sørøst-Norge

Løsningsforslag DAT1000 Databaser 1, vår 2020

Oppgave 1



Oppgave 2

Det finnes mange forskjellige og 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 vurdering.

2-a

Dette spørsmålet er ment å teste **SELECT**-spørring mot én tabell med bruk av sortering og sammensatte betingelser (**AND**).

Jeg valgte å lagre starttidspunkt for presentasjonene med datatype **DATETIME** bestående av både en dato og et klokkeslett. Jeg bruker da **TIME** for å trekke ut klokkeslettet og **DATE** for å trekke ut datoen. Hvis man har representert starttidspunkt i to kolonner, slipper man dette.

```
SELECT TIME(StartTid) AS KlSlett, Tittel
FROM Presentasjon
WHERE RomNr = 101 AND DATE(StartTid) = '2020-05-04'
ORDER BY Tittel;
```

For de fleste vil denne oppgaven la seg løse med en spørring mot én enkelt tabell der man kombinerer to betingelser med **AND**. 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

Dette spørsmålet tester likekoblinger, gruppering og mengdefunksjoner.

```
SELECT T.TemaNr, T.TemaNavn, COUNT(*) AS AntallDeltakere
FROM Deltakertema AS DT INNER JOIN Tema AS T ON DT.TemaNr = T.TemaNr
GROUP BY T.TemaNr, T.TemaNavn;
```

Eller med **WHERE**-betingelse i stedet for **INNER JOIN**:

```
SELECT T.TemaNr, T.TemaNavn, COUNT(*) AS AntallDeltakere
FROM Deltakertema AS DT, Tema AS T
WHERE DT.TemaNr = T.TemaNr
GROUP BY T.TemaNr, T.TemaNavn;
```

2-c

Dette spørsmålet tester likekoblinger og jokernotasjon samt bruk av CONCAT-funksjonen.

SELECT P.Tittel, CONCAT(D.Fornavn, ' ', D.Etternavn) AS Navn

FROM Deltaker **AS** D, Forfatter **AS** F, Presentasjon **AS** P

WHERE D.DNr = F.DNr

AND F.PresNr = P.PresNr

AND P.Tittel **LIKE** '%approach%';

2-d

Dette spørsmålet tester oppdateringsspørringer.

UPDATE Presentasjon

SET RomNr = 101, StartTid = '2020-05-04 10:30'

WHERE PresNr = 1;

2-e

Dette spørsmålet tester bruk av delspørringer, og her skulle man altså både lage en oppgavetekst (som testet bruk av delspørringer) og deretter løse oppgaven.

Eksempel på oppgavetekst: Lag en SQL-spørring som viser deltakere som er medforfattere på en artikkel, men som ikke presenterer.

Løsningsforslaget bruker to delspørringer med bruk av **IN** og **NOT IN**, som gjør at vi får tak i alle deltakere som er med i Forfatter-tabellen, men ikke i Presentasjon-tabellen.

```
SELECT *
FROM Deltaker AS D
WHERE D.DNr IN (SELECT F.DNr FROM Forfatter AS F)
AND D.DNr NOT IN (SELECT P.DNr FROM Presentasjon AS P);
```

Det kunne vært aktuelt å legge til en kolonne «DeltakerType» i Deltaker-tabellen, som hadde gjort denne spørringen enklere.

2-f

Dette spørsmålet tester bruk av **INSERT**.

```
INSERT INTO
  Deltaker(DNr, Fornavn, Etternavn, EPost, DeltType)
VALUES
  (1, 'Peder', 'Aas',     'paas@gmail.com', 'P');
```

```
INSERT INTO
  MåltidBestilling(DNr, DagNr, Lunch, Middag)
VALUES
  (1, 1, TRUE, FALSE),
  (1, 2, TRUE, FALSE),
  (1, 3, TRUE, TRUE);
```

2-g

Dette spørsmålet tester bruk av views og også litt mer kompleks bruk av gruppering, mengdefunksjoner og «formler», særlig hvis man har laget en normalisert datamodell der beløpene for lunch og middag må regnes ut og summeres. Oppgaven kan nok variere i vanskelighetsgrad avhengig av hvordan datamodellen ser ut, noe man må ta særlig høyde for under vurdering.

```
CREATE VIEW Faktura AS
SELECT D.DNr, D.Fornavn, D.Etternavn, KA.Beløp +
  SUM(MB.Lunch*MP.LunchPris + MB.Middag*MP.MiddagPris) AS Beløp
FROM Deltaker AS D, KonferanseAvgift AS KA,
  MåltidPris AS MP, MåltidBestilling AS MB
WHERE D.DNr = MB.DNr AND MP.DagNr = MB.DagNr
GROUP BY D.DNr, D.Fornavn, D.Etternavn;
```

Oppgave 3

3-a

Vi ser altså på følgende tabell:

Tilbakemelding(Dato, Klokkeslett, RomNr, Tittel, Tema, DNr, Fornavn, Etternavn, Vurdering)

Ulempen med den foreslåtte tabellen er at informasjon om en presentasjon (Tittel og Tema) blir gjentatt for alle vurderinger som blir gitt, og informasjon om en deltaker (Fornavn og Etternavn) blir gjentatt for hver gang han eller hun gir en vurdering (vedkommende kan jo møte opp på flere presentasjoner). Dette skaper redundans (dobbeltlagring) og medfører at oppdatering av tabellen blir mer komplisert.

Funksjonelle avhengigheter:

- Dato, Klokkeslett, RomNr, DNr -> ALLE KOLONNER
- Dato, Klokkeslett, RomNr -> Tittel, Tema
- DNr -> Fornavn, Etternavn

Kandidatnøkkel: Dato, Klokkeslett, RomNr, DNr

Den første avhengigheten starter i kandidatnøkkelen, så den kan vi se bort fra.

Avhengigheten fra Dato, Klokkeslett, RomNr bryter med 2NF. Splitting gir en ny tabell som vi kan kalle Presentasjon. Vi fjerner samtidig Tittel og Tema fra Tilbakemelding:

- Tilbakemelding(Dato, Klokkeslett, RomNr, DNr, Fornavn, Etternavn, Vurdering)
- Presentasjon(Dato, Klokkeslett, RomNr, Tittel, Tema)

Den siste avhengigheten, som starter i DNr, bryter også med 2NF. Splitting gir en tredje tabell som vi kaller Deltaker. Fornavn og Etternavn fjernes fra Tilbakemelding. Resultattabellene blir da slik:

- Deltaker(<u>DNr</u>, Fornavn, Etternavn)
- Presentasjon(<u>Dato, Klokkeslett, RomNr</u>, Tittel, Tema)
- Tilbakemelding(<u>Dato*, Klokkeslett*, RomNr*, DNr*</u>, Vurdering)

Primærnøkler er understreket og fremmednøkler er merket med stjerne.

3-b

Verdien til uttrykkene blir bestemt av verdiene til RomNr og Vurdering. Det holder å se på fire tilfeller:

- RomNr=101, Vurdering>5
- RomNr=101, Vurdering<=5
- RomNr<>101, Vurdering>5
- RomNr<>101, Vurdering<=5

I alle fire tilfeller, blir verdiene til de to uttrykkene det samme.

Fordi SQL er basert på treverdilogikk, så burde vi også sett på tilfellene der RomNr og/eller Vurdering inneholder et nullmerke, men både argumentasjon og konklusjon blir den samme.

3-c

Den viktigste grunnen til å opprette primærnøkler og fremmednøkler er at det bidrar til økt datakvalitet. DBHS vil hele tiden sjekke at kravene til slike nøkler, som betyr at vi unngår «meningsløse data», f.eks. to varer med samme varenummer, eller salg av varer vi ikke har.

En bieffekt av fremmednøkler er at de kan utnyttes i forbindelse med applikasjonsutvikling, ved at man mer eller mindre automatisk kan lage skjemaer der brukeren kan velge i lister heller enn å måtte skrive inn verdier for hånd.

3-d

Det er flere fordeler med å lagre data i en database. Noen viktige punkter:

- Data ligger ett sted, det er lettere å holde oversikten.
- Flere kan jobbe med databasen samtidig (og det er støtte for transaksjoner – ikke pensum).
- Ulike brukere kan få ulike tilganger til ulike deler av databasen.
- Man kan bruke spørrespråk som SQL mot databasen.
- Databaser er gode til å søke effektivt i store datamengder.
- Databaser har gode mekanismer for å sikre seg mot feil (backup, loggføring – ikke pensum).

Det er uansett enkelt å eksportere/avlese data fra en database til Excel, for analyse og/eller presentasjon.

Man kan også tenke seg anvendelser der det er enklere å ha data i regnearkfiler, f.eks. hvis dataene bare skal brukes av én person og man bare skal bruke dataene for å lage diagrammer?

3-e

Spørringen viser gjennomsnittlig vurdering for hver kombinasjon av romnummer og tema.

Her bør man sette opp en tabell med noen konkrete eksempelrader og tilhørende spørreresultat. Resultatet vil ha én rad for hver kombinasjon av romnummer og tema.