Arbeidskrav1

```
create table borettslag (
        id int primary key not null,
        navn varchar(255) not null,
        adresse varchar(255) not null,
        etableringsaar int not null
)

create table andelseier (
        id int primary key not null,
        etternavn varchar(255) not null,
        fornavn varchar(255) not null,
        borettslag_id int foreign key references borettslag(id)
)

create table blokk (
        id int primary key not null,
        adresse varchar(255) not null
)

create table etasje (
        id int primay key not null,
        blokk_id int foreign key references blokk(id) not null
)

create table leilighet(
        id int primary key not null,
        etasje_id int foreign key references etasje(id) not null,
        andelseier_id int foreign key references andeliseier(id) not null
)
```

Arbeidskrav2

Oppgave 1

a)

```sql
select fornavn, etternavn
from forfatter
order by etternavn;
```

b)

```sql
select forlag.forlag_id
from forlag
left join bok
on forlag.forlag_id = bok.forlag_id
where bok_id IS NULL;
```

c)

```sql
select * from forfatter
where fode_aar < 1900;
```

d)

```sql
select forlag.forlag_navn, forlag.adresse
from bok
join forlag
on bok.forlag_id = forlag.forlag_id
where bok.tittel = 'sult';
```

e)

```sql
select tittel
from bok
join bok_forfatter
on bok.bok_id = bok_forfatter.bok_id
join forfatter
on bok_forfatter.forfatter_id = forfatter.forfatter_id
where forfatter.etternavn = 'Hamsund';
```

f)

```sql
select concat('tittel: ', bok.tittel, ', utgivelse: ', bok.utgitt_aar) as
'bokInfo',
       concat('forlag: ', forlag.forlag_navn, ', ', forlag.adresse, ', ',
forlag.telefon) as 'forlagInfo'
from bok
join forlag
on bok.forlag_id = forlag.forlag_id;
```

Oppgave 2

a)

```sql
select goal.teamid, goal.player, goal.gtime, game.mdate
from goal
join game
on goal.matchid = game.id
where teamid = 'GER';
```

b)

```sql
select goal.player, count(goal.player) as goalCount
from goal
where teamid = 'GER'
group by goal.player
order by goalCount desc ;
```

c)

```sql
select goal.player, goal.teamid, eteam.coach, goal.gtime
from goal
join eteam
on goal.teamid = eteam.id
where gtime <= 10;
```

d)

```sql
select game.mdate, eteam.coach
from game
join eteam
on game.team1 = eteam.id
where eteam.coach = 'Fernando Santos';
```

e)

```
select game.mdate, eteam.coach
from game
join eteam
on game.team1 = eteam.id
or game.team2 = eteam.id
where eteam.coach = 'Fernando Santos';
```

Arbeidskrav3

Oppgave 1

1.1

```
select * from borettslag
where etabl_aar > 1975 and etabl_aar > 1985;
```

1.2

```
select
    concat(andelseier.fornavn, ' ', andelseier.etternavn, ', ansiennitet:
', andelseier.ansiennitet, ' år') as info
from andelseier
order by ansiennitet desc;
```

1.3

```
select min(borettslag.etabl_aar)
from borettslag
```

1.4

```
select bygning.bygn_adr
from leilighet
join bygning
on leilighet.bygn_id = bygning.bygn_id
where ant_rom >= 3;
```

1.5

```
select * from bygning
where bolag_navn = 'Tertitten';
```

1.6

```
select borettslag.bolag_navn, count(bygn_id) from borettslag
left join bygning
on borettslag.bolag_navn = bygning.bolag_navn
group by borettslag.bolag_navn;
```

1.7

```
select borettslag.bolag_navn, count(leilighet.leil_nr)
from borettslag
join bygning
on borettslag.bolag_navn = bygning.bolag_navn
join leilighet
on bygning.bygn_id = leilighet.bygn_id
group by bygning.bolag_navn;
```

1.8

```
select borettslag.bolag_navn, count(leilighet.leil_nr)
from borettslag
join bygning
on borettslag.bolag_navn = bygning.bolag_navn
join leilighet
on bygning.bygn_id = leilighet.bygn_id
group by bygning.bolag_navn;
```

1.9

```
select andelseier.fornavn, andelseier.etternavn, andelseier.telefon from
andelseier
left join leilighet
on andelseier.and_eier_nr = leilighet.and_eier_nr
where leilighet.and_eier_nr is null;
```

1.10

```
select borettslag.bolag_navn, count(andelseier.and_eier_nr)
from borettslag
left join andelseier
on borettslag.bolag_navn = andelseier.bolag_navn
group by borettslag.bolag_navn;
```

1.11

```
select andelseier.*, leilighet.leil_nr from andelseier
left join leilighet
on andelseier.and_eier_nr = leilighet.and_eier_nr
```

1.12

```sql
select borettslag.bolag_navn
from borettslag
join bygning
on borettslag.bolag_navn = bygning.bolag_navn
join leilighet
on bygning.bygn_id = leilighet.bygn_id
where leilighet.ant_rom = 4;
```

1.13

```sql
select poststed.postnr, count(andelseier.and_eier_nr) as ant_andelseiere
from poststed
left join borettslag
on poststed.postnr = borettslag.postnr
left join andelseier
on borettslag.bolag_navn = andelseier.bolag_navn
group by poststed.postnr
order by ant_andelseiere desc;
```

Oppgave 2

a)

```sql
select *
from ordrehode
join ordredetalj
on ordrehode.ordrenr = ordredetalj.ordrenr
join levinfo
on ordrehode.levnr = levinfo.levnr
where levinfo.levnr = 44;
```

b)

```sql
select levinfo.navn, levinfo.levby
from levinfo
join prisinfo
on levinfo.levnr = prisinfo.levnr
where delnr = 1;
```

c)

```sql
select levinfo.postnr, levinfo.navn, min(prisinfo.pris)
from prisinfo
join levinfo
```

```
on prisinfo.levnr = levinfo.levnr
where prisinfo.delnr = 201;
```

d)

```
select
    ordrehode.ordrenr,
    ordrehode.dato,
    ordredetalj.delnr,
    ordredetalj.kvantum,
    delinfo.beskrivelse,
    prisinfo.pris as enhetspris,
    (prisinfo.pris * ordredetalj.kvantum) as totalpris
from ordrehode
join ordredetalj
on ordrehode.ordrenr = ordredetalj.ordrenr
join delinfo
on ordredetalj.delnr = delinfo.delnr
join prisinfo
on delinfo.delnr = prisinfo.delnr
where ordrehode.ordrenr = 16;
```

e)

```
select
    prisinfo.delnr,
    prisinfo.pris,
    prisinfo.levnr
from prisinfo
where prisinfo.pris > (
    select prisinfo.pris
    from prisinfo
    where katalognr = 'X7770'
    );
```

f)

```
create view syn as
    select *
    from levinfo
where levby = 'Ås';
```

g)

```
select levby
from levinfo as total_lev_info
```

```sql
where not exists(
    select 1
    from prisinfo
    join levinfo as prisinfo_levinfo
    on prisinfo.levnr = prisinfo_levinfo.levnr
    where total_lev_info.levby = prisinfo_levinfo.levby
    );
    ```

h)
```sql
select l.navn, l.levby, SUM(kvantum * pris) as totalPris
from ordredetalj as detalj
    join prisinfo pinfo on detalj.delnr = pinfo.delnr
    join levinfo l on pinfo.levnr = l.levnr
where detalj.ordrenr = 18
group by l.navn, l.levby
order by count(pinfo.delnr) desc, totalPris
limit 1;
```

```sql
create view can_deliver as
select l.levnr, l.navn, p.delnr, p.pris, o.kvantum
from levinfo l
join prisinfo p on l.levnr = p.levnr
join ordredetalj o on p.delnr = o.delnr
where o.ordrenr = 18;

select c.levnr, c.navn, sum(c.pris * c.kvantum) el_pris
from can_deliver c
group by c.levnr, c.navn
having count(*) = (
    select count(*)
    from ordredetalj
    where ordredetalj.ordrenr = 18
    )
order by el_pris
limit 1;
```

Arbeidskrav4

```
leieforhold(kunde_id, kunde_navn, kunde_adresse, kunde_tlf, eiendom_id,
eiendom_adresse, eier_id, eier_navn, eier_adresse, eier_tlf, fra_uke,
til_uke, pris)
```

```
create table kunde(
        kunde_id int primary key,
        kunde_navn varchar,
        kunde_adresse varchar,
        kunde_tlf int
)

create table eiendom(
        eiendom_id int primary key
        eiendom_adresse varchar,
        eier_id int,
        foreign key eier_id references eier(eier_id)
)

create table eier(
        eier_id int primary key,
        eier_navn varchar,
        eier_adesse varchar,
        eier_tlf int
)

create table leieforhold(
        eiendom_id int primary key,
        fra_uke int primary key,
        til_uke int primary key,
        kunde_id int,
        foreign key kunde_id references kunde(kunde_id)
)
```

Teori

Database normalisering

1NF

1. Ikke tillat å bruke radrekkefølge for å formidle data.
2. Ikke tillat å blande datatyper innen samme kolonne.
3. Ikke tillatt å ha tavler uten en primærnøkkel.

4. Repeterende grupper er ikke tillatt.

2NF

Hver non-key attributt må være avhengig av hele primærnøkkelen.

3NF

Hvert ikke-nøkkel attributt må være avhengig av primærnøkkelen, hele primærnøkklen og ingenting annet enn primærnøkkelen.

BCNF

Hvert attributt må være avhengig av primærnøkkelen, hele primærnøkklen og ingenting annet enn primærnøkkelen.

4NF

Flerverdiede avhengigheter i en tavle må være flerverdiede avhegnigheter på nøkkelen.

5NF

En tavle som er i 4NF kan ikke bli beskrevet som et logisk resultat som en union av andre tavler.

Låser

Låsing er en prosedyre for å kontrollere samtidig tilgang til data. Når en transaksjon får tilgang til data, settes en lås som begrenser andre transaksjoners adgang til disse dataene.

Delt lås (S):

Brukes kun ved lesing av data og godtar at flere transaksjoner leser de samme dataene samtidig. En delt lås kan settes selv om andre transaksjoner allerede har satt delt lås på samme data.

Eklusliv lås (X):

Kan kun settes av én transaksjon om gangen. Andre transaksjoner har ikke tilgang til dataene før låsen oppheves. Det hindrer at en transaksjon oppdaterer data mens nadre enten leser eller oppdaterer de samme dataene. En eksklusiv lås kan ikke settes hvis det finnes andre låser, enten delte eller andre eksklusive, på dataene fra før. En eksklusiv lås krever altså at ingen andre transaksjoner leser eller skriver på dataene i det øyeblikket den settes.

Isolasjonsnivåer

Fire isolasjonsnivåer innenfor SQL:

1. READ UNCOMMITED
2. READ COMMITED
3. REPEATABLE READ
4. SERIALIZABLE

Man vil helst ha et isolasjonsnivå lavere enn serializable fordi serializable gjør transaksjonen veldig tung.

Om to pågående transaksjoner med isolasjonsnivå serializable prøver å kjøre
`sql select sum(saldo) from konto`
så vil den første transaksjonen utføres før den andre begynnes.

To-fase låsing (2PL) er et kontroll mekanisme som opererer på prinsippet at transaksjoner anskaffe låsen på dataen før de kan lese eller modifisere dataen.

Det er tre samtidsproblemer som kan oppstå:

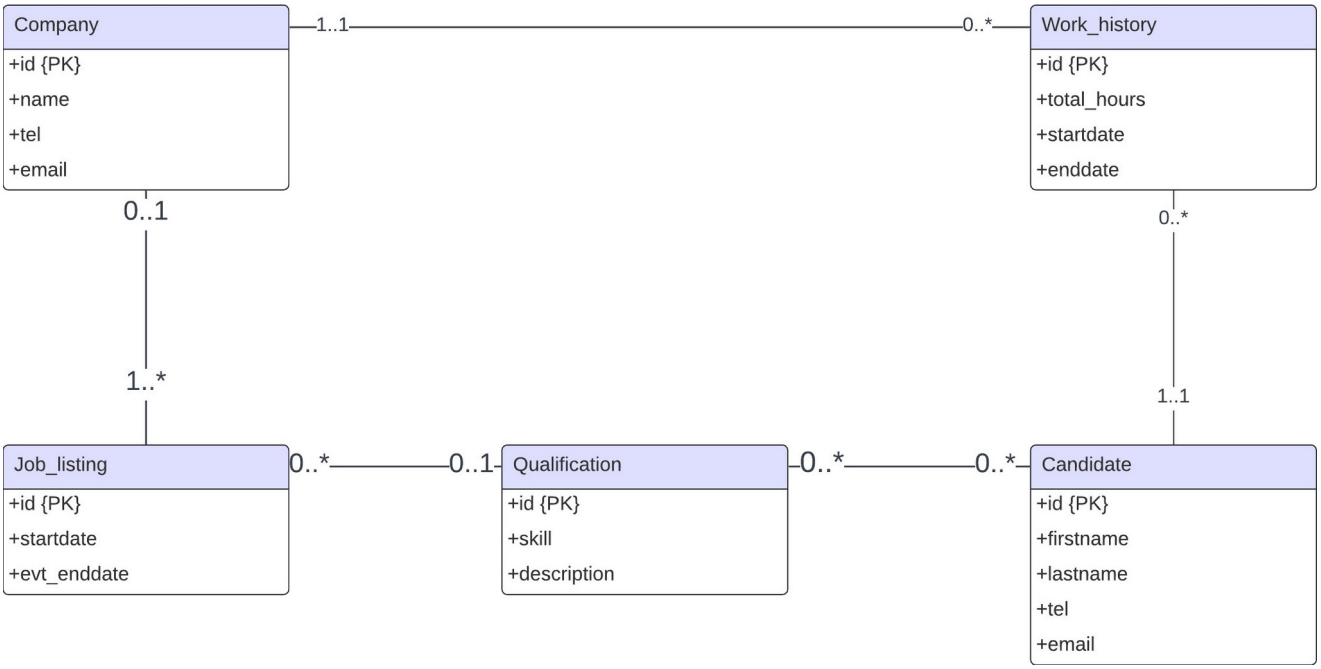
1. Dirty read
2. Nonrepeatable read
3. Phantoms

Optimistisk låsing går ut på å ikke sette en lås på data som leses eller modifiseres. Dette gjøres fordi det er veldig effektivt å utføre. Nyttig i systemer med flest read-operasjoner eller i ikke-kritiske dataoperasjoner

Det kan være dumt med transaksjoner som tar lang tid fordi det vil gjøre lesing fra databasen treigere for alle andre. Bruker-input blir på lik vis ekstremt ineffektivt fordi mange mange transaksjoner kunne ha foregått mens en bruker driver å taster.

Arbeidskrav5

a)



b)

```
Company(
        _id_ int not null,
        name varchar,
        tel int,
        email varchar
)

Job_listing(
        _id_ int not null,
        company_id* int not null,
        qualification_id* int,
        startdate date not null,
        evt_enddate date,
)

Qualification(
        _id_ int not null
        skill varchar,
        description varchar
)

Candidate_Qualification(
        _candidate_id_* int not null,
        _qualification_id_* int not null
```

```
)

Candidate(
        _id_ int not null,
        firstname varchar,
        lastname varchar,
        tel int,
        email varchar
)

Work_history(
        _id_ int not null,
        candidate_id* int not null,
        company_id* int not null,
        total_hours int,
        startdate date,
        enddate date
)
```