Oppgave 1

a) For det første står DML for "Data Manipulation Language". Det er et SQL-språk som ~~oppd~~ kan brukes til å oppdatere kolonner i en tabell i en database med verdier.
For at endringene skal tre i kraft må DBMS få en verifikasjon på at endringen skal skje. Dette gjøres ved å bruke commit-kommandoen etter utsagnet.

```
update table PERSON
set adresse = 'Norgeveien 5'
where personId = 3
commit;
```

Om en angrer på endringene kan man legge på er rollback-kommando istedet:

```
rollback;
```

b) En en-til-mange ~~ret~~ sammenhenger i en ~~Opp~~ relasjonell datamodell er implementert ved at primærnøkkelen i en-sammenhengen blir fremmednøkkelen i mange-sammenhengen. Dette kan illustreres slik:

BYGNING(bygningNr, farge, type);
ROM(romNr, størrelse, bygning_Nr);

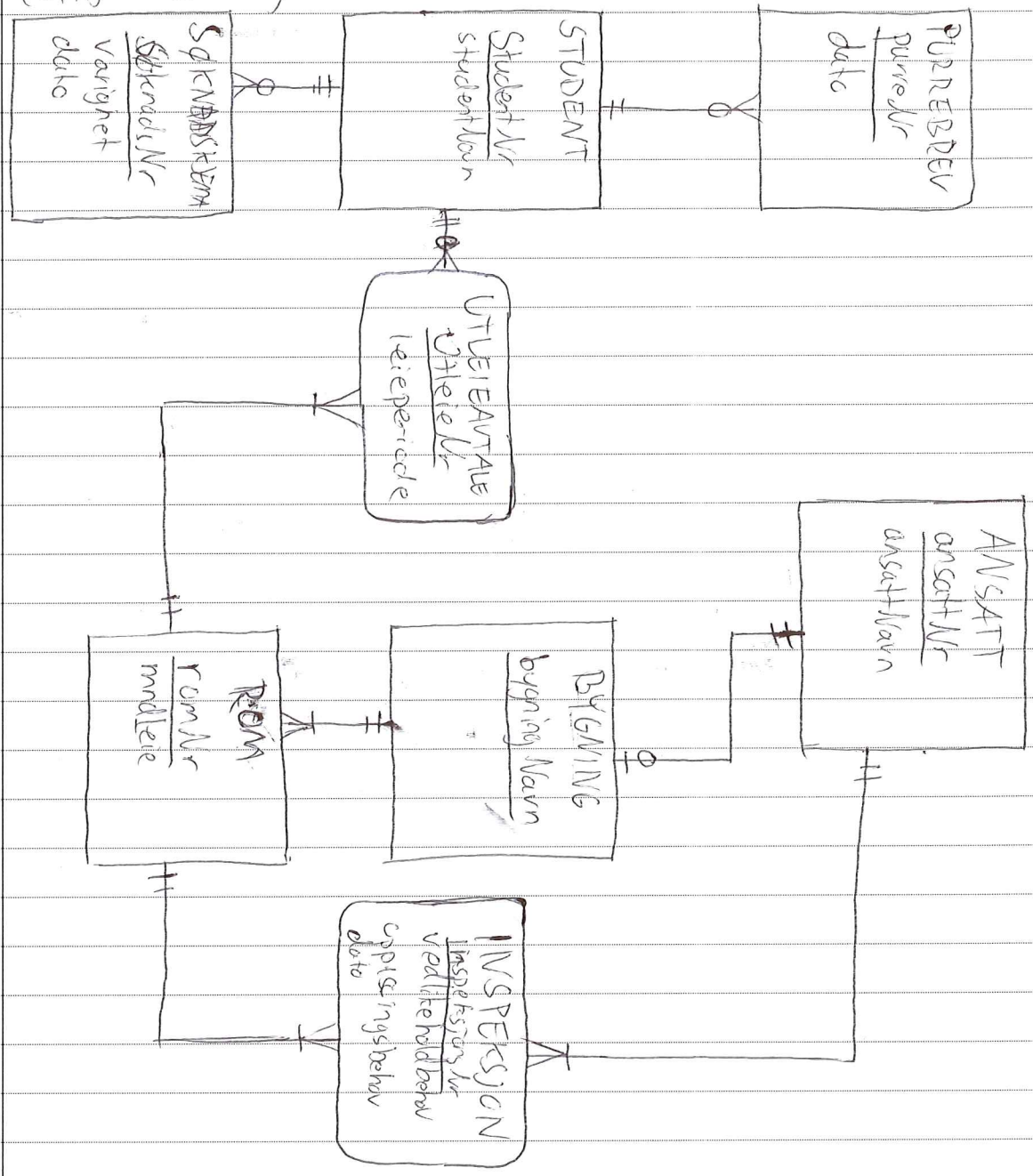
(oppgave 1 forts.)

c) En indeks er en databasestruktur som gjerne brukes i store databaser. Dette er gjerne for at spørringer skal gå kjappere, da indekserte deler svarer spørringer veldig kjapt. Indekser brukes ofte hvis kolonne kan ha mange forskjellige verdier, og blir lagt mest vekt på til de delene av databasen der det blir sendt flest spørringer.
Indekser blir også helst brukt der det er mye større trafikk fra spørringer enn det er fra oppdateringer, da indeksering ikke er til noen særlig nytte for oppdateringer.

Oppgave 2)

(Begynner på neste side)

(oppgave 2 forts.)



ANSATT-entiteten kunne godt bare vært et attributt ~~attributt~~ i BYGNING-entiteten, men jeg valgte å lage en ANSATT-entitet likevel for å holde modellen ryddig. Selv om det ikke er direkte spesifisert i oppgaveteksten tok jeg som utgangspunkt at Boligkontoret har som interesse å lagre data om sine ansatte.

(Oppgave 2 forts.)
Jeg gav også alle entititene en primærnøkkel de det ikke var spesifisert. Dette for å holde konsistens i modellen

Oppgave 3)

Denne rapporten er i 0.NF. Dette er fordi den inneholder repeterende grupper/multi-valued attributter, og det er heller ikke enda spesifisert noen primærnøkkel

Dette er de funksjonelle avhengighetene:

WardNo → Doctor-in-charge, phone ~~date~~
Nurse → phone, address
Patient-No → Last_Name, first_name, home_City, visit_Date, diagnosis, reffeedTo

For å få tabellen til 1NF må vi spesifisere primærnøkkelen i relasjonen:

REPORT(WardNo, doctor-in-charge, doctor_phone, date, nurse, nurse_phone, nurse_address, patient_No, last_name, first_name, home_city, visit_date, diagnosis, reffered_To),

(oppgave 3 forts.)

For å få ~~tabellen~~ relasjonen videre til 2NF må vi fjerne de (på engelsk) partial functional dependency'ene. En partial dependency er når en eller flere attributter er avhengig av en del av primærnøkkelen fra relasjonen i 1NF. For hver av disse avhengighetene lager vi en ny relasjon med ~~den~~ delen av primærnøkkelen som primærnøkkel i den nye relasjonen. Deretter flytter vi alle attributter som er avhengig av denne nøkkelen over i den nye relasjone.

2NF:
REPORT (Ward_No, ~~Nurse~~ nurse_name, patient_No, date);
WARD (Ward_No, doctor-in-charge, doctor_phone);
NURSE (nurse_name, nurse_phone, nurse_address);
PATIENT (patient_No, last_name, first_name, home_city, visit_date, diagnosis, referedTo);

For å få relasjonen videre til 3NF må vi også fjerne de transitive avhengighetene. En transitiv avhengighet er når en eller flere attributter er avhengig av en annen non-key attributt. For hver non-key attributt med transitiv avhengighet lager vi en ny relasjon med denne som primær-nøkkel og ~~flytt~~ flytter attributtene som er avhengig av denne over i den nye relasjonen. Primærnøkkelen i den nye relasjonen blir stående som en fremmednøkkel i den gamle relasjonen.

(Oppgave 3 forts.)

Med dette står vi igjen med disse relasjonene i 3NF:

REPORT(WardNo, nurse_name, patient_No, date);
WARD(WardNo, doctor_in_charge);
NURSE(nurse_name, nurse_phone, nurse_address);
PATIENT(patient_No, last_name, first_name, home_city, visit_date, diagnosis, reffered to);
DOCTOR(doctor_name, doctor_phone);

Oppgave 4

a) Constraints er regler som sier noe om åssen data skal "oppføre" seg i en database. Dette kan f.eks være en integrity constraint, som er en regel som sier at en primærnøkkel må være unik og kan aldri være null. Eller det kan være en referential constraint som sier at fremmednøkler kan være null, men må peke til en primærnøkkel om den ikke skal være null.
For å opprette constraints til ordredetalj, så kan vi ~~endre~~ oppdatere tabellen med ALTER-klausulen:

```
ALTER TABLE ordredetalj(
add constraint ord_det_pk primary key (ordrenr, produktnr),
add constraint ordre_fk foreign key (ordrenr) references
    ordre(ordrenr),
add constraint produkt_fk foreign key (produktnr) references
    produkt(produktnr));
```

b)

```
SELECT   produktnr, produktnavn
FROM     produkt
INNER JOIN ordredetalj
ON ordredetalj.produktnr = produkt.produktnr
INNER JOIN ordre
ON ordre.ordrenr = ordredetalj.ordrenr
WHERE ordre_dato LIKE '2010-10%'
ORDER by produktnavn;
```

c)

```
SELECT kundenr, kunde_navn
FROM kunde
WHERE kundenr NOT IN
   ( SELECT kundenr
     FROM kunde
     INNER JOIN ordre
     ON ordre.kundenr = kunde.kundenr);
```

d)

```
SELECT kundenr, kundenavn, sum(antall_bestilt) TotBestilt,
       sum(antall_bestilt * enhetspris) TotalPris
FROM   kunde
INNER JOIN ordre ON ordre.kundenr = kunde.kundenr
INNER JOIN ordredetalj ON ordredetalj.ordrenr = ordre.ordrenr
INNER JOIN produkt ON produkt.produktnr = ordredetalj.produktnr
group by kundenr, kundenavn;
```

fra d)

e) Da ender vi på spørringen fra etter siste INNER JOIN:

```
WHERE kunde.postnr BETWEEN 4600 AND 4699
GROUP BY kundenr, kundenavn
    HAVING TotBestillt > 5
;
```

f) Et view er en slags virtuell tabell der man kan spesifisere vetiry detaljert hvilke kolonner og attributter man vil ha med, fra eventuelt flere tabeller.
Vi bruker views grunnet flere ting. Et view kan gjøre det lettere å kjøre spørringer, man trenger ikke koble sammen tabeller hele tiden.
Man kan også gi forskjellig brukere adgang til forskjellige views. Dette beder adgangskontroll og samtidig sikkerheten blant dataene.

Et enkelt view som lister alle kunder som har ordre, og ordren.

```
CREATE VIEW kundeOrdre AS (
  select kunde_navn, ordrenr
  from kunde
  INNER JOIN ordre
  ON kunde.kundenr = ordre.kundenr);
```