EKSAMEN TDAT1005 MAI 2019
Løsningsforslag/sensurveiledning

OPPGAVE 1

Om sensur

EER/ER-modellen og relasjonsmodellen må henge sammen. Oversettingen fra den ene til den andre må være riktig.

Dersom kun en av modellene eksisterer for en del av oppgaven gis maks halv uttelling for denne delen.

Ståkarakter krever at noen få entitetstyper med sammenhengstyper er modellert riktig, eksempelvis må en en-til-mange-sammenheng settes opp «riktig vei». Oversetting til relasjonsmodellen må også være riktig for en liten del av oppgaven. Dette er minimumskravene for ståkarakter, jmf definisjonen på karakteren E.

Topp-karakter krever (bl.a.) at elementer fra EER-modelleringen er tatt i bruk, i praksis betyr dette at spesialisering og generalisering er vist i figuren.

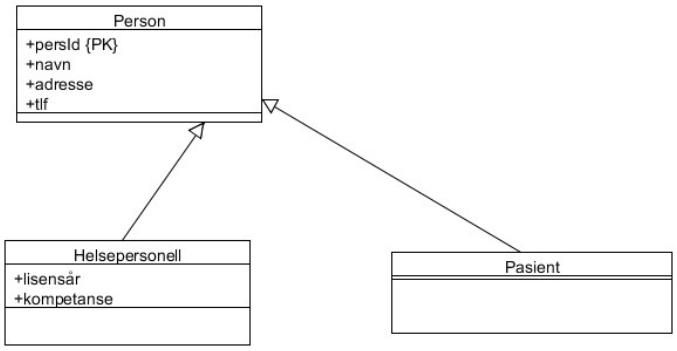
Primærnøkler er understreket i relasjonsmodellen nedenfor. Studentene må bruke understreking før og etter (eksempel _persId_ i stedet for persId) pga at Inspera ikke har støtte for vanlig understreking for denne oppgavetypen.

Eventuelle alternative nøkler er kun markert i diagrammet nedenfor, og da med {UNIQUE}. Dette er ikke en «syntaks» som kreves i besvarelsene.

Ang oversetting av arv (generalisering/spesialisering)

Det henvises til læreboka i databaser side 206-207.

Det er spesielt to misforståelser som går igjen hos noen av studentene. Jeg viser det ved eksemplet med person, helsepersonell og pasient som er en del av oppgaven:



MISFORSTÅELSE 1:

Man gjentar alle attributter i subentitetstypene. Da ser det slik ut:

PERSON(persId, navn, adresse, tlf)
HELSEPERSONELL(helsePersId*, navn, adresse, tlf, lisensår, kompetanse)
PASIENT(pasientPersId*, navn, adresse, tlf)

Her er helsePersId og pasientPersId fremmednøkler som refererer til persId i relasjonen Person. Dette betyr at informasjon om navn, adresse og telefin lagres to ganger for hver person, en gang i PERSON og en gang, entet i HELSEPERSONELL eller i PASIENT.

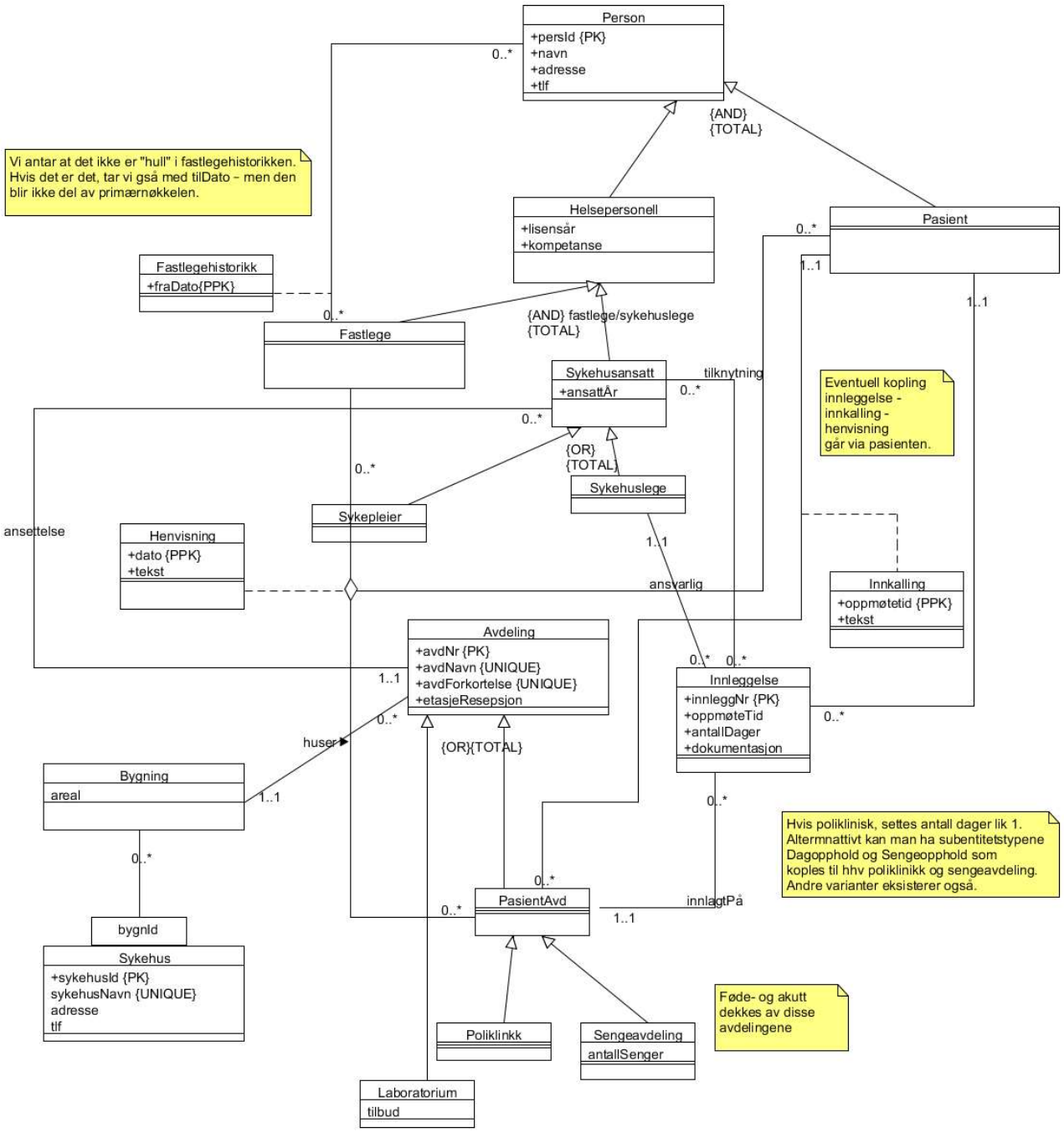
Denne misforståelsen anses som en *grov feil*.

MISFORSTÅELSE 2:

Denne er som over, men relasjonen Person utelates.

Dette er noe bedre på en måte, ettersom data lagres kun én gang. Men dette er ikke i samsvar med EER-modellen. Prøv og tegne modellen uten entitetstypen Person. Sammenhengstypene som Person inngår i blir nå ikke veldig elegante, de må gå både til Helsepersonell og til Pasient, det blir i det hele tatt en meget dårlig modell.

Oppgave 1 - ER/EER-modell, se neste side



Oppgave 1 – relasjonsmodell

SYKEHUS(sykehusId, sykehusnavn, adresse, tlf)
BYGNING(sykehusId*, bygnId, areal)

AVDELING(avdNr, avdNavn, avdForkortelse, etasjeResepsjon, (sykehusId, bygnId)*)
LABORATORIUM(labAvdNr*, tilbud)
PASIENTAVD(pasAvdNr*)

POLIKLINIKK(poliKlinikkAvdNr*)
SENGEAVDELING(sengeAvdNr*, antSenger)

PERSON(persId, navn, adresse, tlf)
HELSEPERSONELL(helsePersId*, lisensår, kompetanse)
SYKEPLEIER(sykeplPersId*)
SYKEHUSLEGE(sykehusLegePersId*)
PASIENT(pasientPersId*)

FASTLEGE(fastlege helsePersId*)
FASTLEGEHISTORIKK(persId*,fastlege helsePersId*, fraDato)

HENVISNING(pasientPersId*, fastlege helsePersId*, pasAvdNr*, dato, tekst)
INNKALLING(pasientPersId*, pasAvdNr*, oppmøteTid, tekst)
INNLEGGELSE(innleggNr,pasientPersId*, pasAvdNr*, dato, sykehusLegePersId*, antDager, dokumentasjon)
TILKNYTNING(innleggNr*, helsePersId*)

Oppgave 2 - SQL

-- oppg. a

```
SELECT DISTINCT t.typeid, ant_personer
FROM campus c, bygning b, rom r, romtype t
WHERE c.campusid = b.campusid
AND b.bygnnavn = r.bygnnavn
AND r.typeid = t.typeid
AND camp_navn = "Kalvskinnet";
```

-- oppg. b

```
SELECT DISTINCT campusid
FROM bygning b, rom r
WHERE b.bygnnavn = r.bygnnavn
AND etanr = (SELECT MAX(etanr) FROM rom);
```

-- oppg. c

```
SELECT c.campusid, camp_navn, r.typeid, COUNT(romid) AS ant_rom
FROM campus c, bygning b, rom r
WHERE c.campusid = b.campusid
AND b.bygnnavn = r.bygnnavn
```

```
GROUP BY c.campusid, r.typeid
ORDER BY camp_navn;


-- oppg. d
SELECT romid, romnavn
FROM rom
WHERE bygnnavn = "Sverresgate 10"
AND romid NOT IN (
      SELECT r.romid
      FROM reservasjon re, rom r
      WHERE re.romid = r.romid
      AND bygnnavn = "Sverresgate 10"
      AND dato = '2019-06-01'
      AND fra_tid < '10:00' AND til_tid >= '10:00');
```

-- oppg. e

Kan lagre resultatet fra oppg. d som et VIEW:

```
CREATE VIEW ledige_rom
AS SELECT romid, romnavn
FROM rom ...oppg. d;
```

Kan deretter hente MIN(romid) fra view-et ved INSERT i reservasjon:

```
INSERT INTO reservasjon (romid, dato, fra_tid, til_tid)
VALUES((SELECT MIN(romid) FROM ledige_rom),'2019-06-01', '09:00',
'10:00');
```

OPPGAVE 4a – RELASJONSALGEBRA

Oppgaven handler om relasjonsalgebraen, det vil si det *teoretiske grunnlaget* for SQL. SQL-setninger godtas ikke som svar på oppgavene.

Svar på deloppgavene:

i) Seleksjon*)

ii) Naturlig forening**), eventuelt
Venstre ytterforening, eventuelt
(Indre) forening etterfulgt av projeksjon***), eventuelt
Kryssprodukt etterfulgt av seleksjon og projeksjon***)

iii) Som ii) etterfulgt av projeksjon***)

iv) Union, snitt og (mengde-)differanse krever unionkompatibilitet.
PLANTE og PARK er unionkompatible fordi de har like mange attributter og disse er av samme type. (Rekkefølgen på attributtene spiller ingen rolle.)

*) Restriksjon er at annet navn for seleksjon.
**) «Join» godtas i stedet for «forening».
***) Reduksjon er et annet navn for projeksjon.

Oppgave 4c) - TRANSAKSJONER

Vi har fire ulike isolasjonsnivåer: Read uncommitted, read committed, repeatable read og serializable.
Her bør man nevne ulike problemer som kan oppstå i de ulike nivåene (og knytte dem til riktig nivå) som "dirty read", "non-repeatable read" og "phantom reads".
Grunnen til at man ikke nødvendigvis ønsker full isolering
(serializable) er at dette er ressurskrevende og gir vesentlig dårligere ytelse - spesielt om det er mye lesing av data. Man går derfor gjerne for et mindre strengt isoleringsnivå.
Leselåser og skrive låser kan også være naturlig å ta inn i forklaringen (selv om man svarer på oppgaven uten å ta med denne delen).