Oversikt over vedlegg

1. Case-beskrivelsen, slik denne lå i Blackboard.
2. SQL-script for å lage databasen i Oppgave 2 (også gitt på forhånd)

Merk:
For å få ståkarakter i emnet, må kandidaten ha ståkarakter på Oppgave 1, 2 og 3 hver for seg.

B (eller A) forutsetter også ståkarakter på Oppgave 4.

Oppgave 1) Datamodellering – 25%

Dersom case-beskrivelsen dere fikk på forhånd omfatter mer, eller kan tolkes annerledes, enn det som står her, så er det det som står her som gjelder.

Du skal i denne oppgaven modellere en databaseløsning for sykehusene i en region. Du skal modellere sykehusene med bygninger, avdelinger, ansatte, pasienter og innleggelser. På grunn av at også fastlegene er sentrale i denne sammenhengen, skal disse til en viss grad med i modellen. Detaljer følger nedenfor.

Et sykehus identifiseres med et nummer. Det har også et entydig navn. I tillegg lagres adresse og telefon.

Et sykehus består av mange bygninger. En bygning er identifisert med sykehusnummeret og en bokstav. Bygningens areal skal inn i databasen.

Et sykehus har mange avdelinger. En avdeling identifiseres med et nummer, men har også et entydig navn med en entydig forkortelse. Hver avdeling er plassert i én bygning, og det er plass til hele avdelingen i denne bygningen. Det er plass til flere (hele) avdelinger i hver bygning. Informasjon om i hvilken etasje resepsjonen for den enkelte avdelingen holder til, skal inn i databasen.

Avdelingene er av tre typer: Poliklinisk, sengeavdeling og laboratorium. Om sengeavdelingene skal antall senger inn i databasen. Om et laboratorium skal det med en tekst om tilbudet.

Det er mange kategorier ansatte ved et sykehus, men vi begrenser oss til sykepleiere og leger (som også kan være fastleger). En ansatt er knyttet til en avdeling ved et av sykehusene. For enkelthets skyld bruker vi personnummeret som identifikasjon. I tillegg lagres navn, adresse, telefon, samt året vedkommende ble ansatt.

Databasen skal inneholde fastlegene med personnummer (identifikasjon), navn, adresse og telefon. Alle har en fastlege, også fastlegene selv og de ansatte ved sykehusene. En person kan skifte fastlege inntil to ganger pr år. Historikk over kopling fastlege – person skal lagres i databasen.

Alt helsepersonell må ha lisens for å drive sin virksomhet. Året vedkommende fikk sin lisens skal lagres i databasen. Formell kompetanse skal også lagres (som en tekst).

Så til pasientene. Vi begrenser oss til sykehusinnleggelser. Vi ser altså bort fra besøk hos fastlegen i denne oppgaven. I tillegg til personnummeret (som vi bruker som identifikasjon), lagres navn, adresse og telefon.

Ved søk i databasen skal det være mulig å finne informasjon om en pasients kontakt med sykehusene:

- Et sykehusbesøk kan begynne med at en fastlege sender en henvisning til en bestemt avdeling på et sykehus, der hun ber om undersøkelse av en bestemt pasient. Om en henvisning må man kunne hente ut selve henvisningsteksten, datoen henvisningen ble sendt, informasjon om pasient, adressat (sykehusavdeling), og navn på fastlege. Vi antar at det kun er én henvisning pr dato pr pasient og avdeling.
- Som en følge av henvisningen sender sykehuset innkalling til pasienten med dato, tid og sted (dvs sykehus, avdeling og bygning) for oppmøte.
- Sykehuset kan sende innkalling til en pasient som oppfølging av en tidligere konsultasjon. Dette er da uten henvisning fra fastlegen.
- En pasient kan legges inn på akutt- eller fødeavdelingen. Dette er selvfølgelig uten henvisning og innkalling på forhånd.
- Ved oppmøte knyttes pasienten til en ansvarlig sykehuslege. Andre leger og sykepleiere knyttes også til sykehusoppholdet.
- Enten skrives pasienten ut samme dag (poliklinisk konsultasjon), eller han blir på sykehuset en kortere eller lengre periode. Vi regner oppholdet i hele dager.
- For hvert sykehusbesøk skriver ansvarlig lege en tekst som dokumenterer besøket. Vi ser bort fra det mer omfattende begrepet «journal» i denne oppgaven.

Det skal være mulig å hente ut full historikk (innleggelsesdatoer, avdeling, ansvarlig lege og andre involverte ansatte, samt dokumentasjon fra legen) for en pasient.

Oppgave:

Lag en datamodell (ER/EER) for problemstillingen. Bruk UML-notasjon.

Merk at det i enkelte tilfeller kan være fornuftig å benytte et løpenummer som primærnøkkel, selv om det ikke er nevnt i oppgaveteksten.

Husk at primærnøkler alltid skal markeres i datamodellen. Fremmednøkler hører strengt tatt ikke hjemme i denne modellen, men om du ønsker kan du ta dem med. Da må du i tilfelle ta med alle, og du må markere dem med stjerne.

Oversett datamodellen til relasjonsmodellen. Sett opp relasjoner (tabeller) på relasjonell form (slik som tabellene i Oppgave 2 nedenfor er satt opp). Strek under primærnøkler og marker fremmednøkler med stjerne. Om du ikke får til understreking i Inspera, bruk strek foran og bak ordet, slik: *_persNr_*

Oppgave 2) SQL – 20%

Tabellene (relasjonene) under brukes til å registrere romreservasjoner i ulike bygninger i et tenkt forenklet system for et universitet med flere campus.

CAMPUS (<u>campusid</u>, camp_navn)

BYGNING (<u>bygnnavn</u>, campusid*)

ETASJE (bygnnavn*, etanr)

ROM (romid, romnavn, typeid*, (bygnnavn, etanr)*)

ROMTYPE (typeid, beskrivelse, ant_personer)

RESERVASJON (resid, kontakt_epost, romid*, dato, fra_tid, til_tid, kommentar)

Universitetet har flere campus som kan ha flere bygninger med flere etasjer og rom. Et rom er av en bestemt romtype. Hver romtype har en beskrivelse samt et tall for maks. antall personer romtypen har plass til, oppgitt ved attributtet ant_personer. Tabellen ETASJE brukes til å vise hvilke etasjer (etasjenummer) i bygninger det finnes rom for reservasjon i – dette kan variere fra bygning til bygning.

Tabellen RESERVASJON inneholder data for reservasjoner av rom. Vi forenkler litt, og lagrer kun epost-adressen til personen som har foretatt reservasjonen i attributtet kontakt_epost. En reservasjon gjelder for en gitt dato og fra- og til klokkeslett, gitt ved attributtene dato, fra_tid, til_tid.

Se for øvrig vedlegg for SQL-script. I hver deloppgave under skal du skrive en eller flere SQL-spørringer (setninger) for å finne resultatet.

Oppgave 2a)

List ut alle entydige romtyper og antall personer for hver romtype som finnes på campus-et ved navn "Kalvskinnet" (antar her at det finnes kun ett campus i databasen med det campusnavnet).

Oppgave 2b)

Finn ut hvilke campuser, ett evt. flere campus, som har rom i det høyeste etasjenummeret i databasen det er registret rom i. Skriv ut attributtet campusid i svaret.

Oppgave 2c)

Lag en liste som viser antall rom av hver romtype for hvert registrerte campus i databasen. Sorter resultatet alfabetisk på campusnavn.

Oppgave 2d)

Sjekk hvilke rom (romid) som er ledig den 01.06.19 fra kl. 9:00 til kl. 10:00 i bygningen ved navn "Sverresgate 10". Spørringen skal skrive ut romid og romnavn på ledig rom.

Oppgave 2e)

Gitt at det finnes minst ett ledig rom i resultatet fra Oppgave d): skriv en, evt. en sekvens, av SQL-spørring(er) for å reservere det ledige rommet med lavest romid fra resultatet på tidspunktet oppgitt i Oppgave d).

Oppgave 3 Programmering – 40%

Dersom case-beskrivelsen dere fikk på forhånd omfatter mer, eller kan tolkes annerledes, enn det som står her, så er det det som står her som gjelder.

Pass på at du ikke gjør verken mer eller mindre enn det oppgavene spør etter. Men dersom du trenger flere metoder/konstruktører for å lage det oppgavene spør etter, skal du også

programmere disse. En fornuftig oppdeling i metoder ut over det oppgaven spør etter, kan gi plusspoeng ved bedømmelsen.

Husk at du for alle metoder må oppgi hvilken klasse metoden tilhører.

Du trenger ikke sette opp import-setninger.

I programmeringsoppgaven skal du jobbe med kun ett sykehus med avdelinger, ansatte og pasienter. Du skal ikke programmere mot databasen i denne oppgaven.

Oppgave 3a)

Denne deloppgaven består av å tegne et UML-klassediagram som dekker alle klassene i denne oppgaven. Begynn med en skisse nå, og fintegn diagrammet når du er ferdig med alle deloppgavene. Pass på at diagrammet stemmer overens med det du har programmert. Diagrammet skal dekke alle elementene som benyttes i deloppgavene.

Oppgave 3b)

Du skal nå programmere klassene for personer knyttet til sykehuset. Vi begrenser oss til ansatte (leger og sykepleiere) og pasienter. Klassetreet skal altså bestå av klassene Person, Ansatt, Lege, Sykepleier og Pasient. (Merk at fastlegene ikke er med her, slik de var i oppgave 1.)

Gitt følgende:

- Ansatt er en Person
- Lege og Sykepleier er Ansatt
- Pasient er en Person.
- Følgende informasjon skal lagres for klassene:
    - Person: fornavn, etternavn og personnummer. Klasse skal ha metoder for å hente og sette disse attributtene
    - Pasient: diagnose. Kun klassen «Lege» skal kunne sette diagnose.

- Person: fornavn, etternavn og personnummer. Klasse skal ha metoder for å hente og sette disse attributtene
- Pasient: diagnose. Kun klassen «Lege» skal kunne sette diagnose.

Oppgave 3c)

Klassen «Avdeling» skal ha følgende attributter: avdelingsnavn, ansatte (ArrayList) og pasienter (ArrayList). Listene skal være dynamiske, dvs. programmet utvider listene etter behov. Definer klassen Avdeling.

Definer alle get- og set metodene for objektvariablene.

Metoden «equals» hører til klassen Object som er arvet av klassen Avdeling. I siste deloppgave (oppgave e) skal vi sortere alle avdelingene i et sykehus etter avdelingsnavn.

```
@Override
 public boolean equals(Object o) {
…
  }
```

I denne deloppgaven skal du override equals-metoden slik at den returnerer true eller false avhengig av om 2 avdelinger har like navn eller ikke.

Oppgave 3d)

Et sykehus har et navn og en liste med Avdelinger (ArrayList). Klassen Sykehus skal implementere en grensesnitt-klasse (et interface) ISykehus. Denne har to abstrakte metoder *registrer()* og *fjern()*. Begge metodene tar imot et object (Klassen Object). Metodene returnerer true kun dersom objektet som sendes inn blir registrert/fjernet, ellers returnerer den false.

Metoden registrer() skal sjekke om objektet som kommer inn er av typen Avdeling. I så fall skal denne registreres i listen over avdelinger i klassen Sykehus.

Metoden fjern() skal sjekke om objektet er av typen Avdeling, Pasient eller Ansatt og fjerne dette.

Definer grensesnittklassen ISykehus og klassen Sykehus.

Implementer metodene registrer() og fjern() i klassen Sykehus.

Oppgave 3e)

Du skal nå lage en klasse for å sortere avdelinger basert på antall ansatte i stigende rekkefølge.

1) Lag en komparator-klasse (implementerer Comparator) som skal brukes til å sortere avdelinger avhengig av antall ansatte som er registrert. Kall klassen «AvdSortKomparator»

2) Metoden «compare» skal overrides i klassen AvdSortKomparator. Skriv denne metoden.

3) Gitt følgende klientklasse:

```
public class SykehusKlient {


    public static void main(String[] args) {


        Sykehus sykehus = new Sykehus("St. Olav");


        //Opprett fødeavdeling
              Avdeling fødeAvdeling = new Avdeling("Fødeavdeling");


              Lege lege1 =new Lege("Odd Even","Primtallet","20108044221");
        fødeAvdeling.getAnsatte().add(lege1);


              Lege lege2 =new Lege("Huppasahn","DelFinito","30018041222");
        fødeAvdeling.getAnsatte().add(lege2);


              Sykepleier sykepleier =new Sykepleier("Rigmor","Mortis","12048021456");
```

```
fødeAvdeling.getAnsatte().add(sykepleier);

	Ansatt ansatt = new Ansatt("Kurt","Isere","24119041756");
fødeAvdeling.getAnsatte().add(ansatt);


	Pasient pasient = new Pasient("Horst","Nuppentaler", "11059352706");
lege1.setDiagnose(pasient, "Forkjølet");
fødeAvdeling.getPasienter().add(pasient);

sykehus.registrer(fødeAvdeling);

//Opprett lungeavdeling
	Avdeling lungeAvdeling = new Avdeling("Lungemedisinsk avdeling");

	Lege lege3 =new Lege("Salti","Kaffen", "02037352992");
lungeAvdeling.getAnsatte().add(lege3);

	Sykepleier sykepleier2 =new Sykepleier("Anton","Nym","12098852990");
lungeAvdeling.getAnsatte().add(sykepleier2);

	Ansatt ansatt2 = new Ansatt("Sesam","Lukkopphagen", "10037852123");
lungeAvdeling.getAnsatte().add(ansatt2);


	Pasient pasient2 = new Pasient("Gene","Sis","10067967732");
lege3.setDiagnose(pasient, "Kalle føtter");
lungeAvdeling.getPasienter().add(pasient2);

sykehus.registrer(lungeAvdeling);

//Sortering
…
 }
```

}

Bruk klassen AvdSortKomparator og skriv koden etter kommentaren «//Sortering» ovenfor til å sortere avdelinger.

Oppgave 4 Teori – 15%

Oppgave 4a) – RELASJONSALGEBRA

Følgende tre relasjoner med eksempeldata er gitt (primærnøkler er understreket, fremmednøkler er merket med stjerne *):

PLANTE(planteNr, plantenavn, antallPåLager)

planteNr	plantenavn, maks 50 tegn	antallPåLager
1	Asters	1000
2	Stemor	2000
3	Tagetes	800
4	Storklokke	1500

PARK(parkNr, parknavn, areal)

parkNr	parknavn, maks 50 tegn	areal, helt antall kvm
1	Byparken	500
2	Ugleparken	200
3	Bekkevollen	1200

UTPLANTING(planteNr*, parkNr*, antall)

planteNr	parkNr	antall
1	2	50
2	2	40
4	1	60
4	3	40

Tabellene viser hvor mange planter av hver type som skal plantes ut i forskjellige parker nå i vår.

Vi har gjennomført relasjonsalgebra på relasjonene. Resultatene ser du nedenfor. For hvert av de tre resultatene skal du angi hvilke(n) algebraoperasjon(er) som er benyttet.

i)

parkNr	parknavn	areal
2	Ugleparken	200

ii)

planteNr	parkNr	antall	parkNavn	areal
1	2	50	Ugleparken	200
2	2	40	Ugleparken	200
4	1	60	Byparken	500
4	3	40	Bekkevollen	1200

iii)

planteNr	antall	parkNavn
1	50	Ugleparken
2	40	Ugleparken
4	60	Byparken
4	40	Bekkevollen

iv)

Hvilke operasjoner krever at operandene er unionkompatible? Er noen av de tre relasjonene gitt i begynnelsen av oppgaven (PLANTE, PARK og UTPLANTING) unionkompatible med hverandre? Begrunn svaret.

Oppgave 4b) - PROGRAMMERING

Gitt koden under:

```
public class ReferanseTest {

private Person testObj;

public ReferanseTest() {

testObj = new Person(1,"Per Person");

}


public Person getPerson() {

return testObj;

}
```

```
class Person {
String lokalNavn;
Integer lokalId;

Person(int id, String navn) {
            lokalId = id;
             lokalNavn = navn;
}

public void setNavn(String str) {
             lokalNavn = str;
}

 public void setId(Integer id) {
           lokalId = id;
}

 public String getNavn() {
           return lokalNavn;
}

 public Integer getId() {
           return lokalId;
}

 public String toString() {
            return lokalId + " " + lokalNavn;
}
 }//end Person

  public static void main(String[] args) {
      ReferanseTest refTest = new ReferanseTest();
```

```
        Person testPerson = refTest.getPerson();

       System.out.println(testPerson);

        Integer id = testPerson.getId();
        String navn = testPerson.getNavn();

        id = 0;
        navn = "Bør Børson";

        testPerson = refTest.getPerson();

        System.out.println(testPerson);

        testPerson.setId(11);
        testPerson.setNavn("Charlie Chaplin");

        testPerson = refTest.getPerson();

      System.out.println(testPerson);

    }//end main
}
```

Hva blir resultatet når man kjører koden? Grunngi svaret.

Oppgave 4c) - TRANSAKSJONER

Beskriv de ulike isolasjonsnivåene vi har i forbindelse databasetransaksjoner. Forklar kort fordeler/ulemper ved de ulike isolasjonsnivåene.

Vedlegg 1: Case-beskrivelsen, slik denne lå i Blackboard

Inspirasjon til caset er hentet fra hjemmesiden til St. Olavs Hospital (https://stolav.no/). Caset er meget forenklet for å tilpasse pensum og tidsrammen for eksamen i dette faget.

Definisjon hentet fra Store Medisinske Leksikon (https://sml.snl.no/sykehus): *Sykehus er en helseinstitusjon som er godkjent for innleggelse, undersøkelse og behandling av pasienter og fødende kvinner som trenger spesialisert helsetjeneste. Til sykehusene hører sengeavdelinger, poliklinikker og laboratorier. Tradisjonelt har man skjelnet mellom somatiske sykehus, som hovedsakelig behandler kroppens sykdommer, og psykiatriske sykehus, som behandler psykiske lidelser. Dette skillet er av mindre betydning nå i og med at psykiatriske avdelinger ofte inngår som en del av det totale sykehustilbudet.*

Oppgave 1) Datamodellering – 25%

Du skal i denne oppgaven modellere en databaseløsning for sykehusene i en region. Du skal modellere sykehusene med bygninger, avdelinger, ansatte, pasienter og innleggelser. På grunn av at også fastlegene er sentrale i denne sammenhengen skal disse til en viss grad med i modellen.

Oppgave 2) SQL – 20%

Tabellene (relasjonene) under brukes til å registrere romreservasjoner i ulike bygninger i et tenkt forenklet system for et universitet med flere campus.

CAMPUS (campusid, camp_navn)

BYGNING (bygnnavn, campusid*)

ETASJE (bygnnavn*, etanr)

ROM (romid, romnavn, typeid*, (bygnnavn, etanr)*)

ROMTYPE (typeid, beskrivelse, ant_personer)

RESERVASJON (resid, kontakt_epost, romid*, dato, fra_tid, til_tid, kommentar)

Universitetet har flere campus som kan ha flere bygninger med flere etasjer og rom. Et rom er av en bestemt romtype. Hver romtype har en beskrivelse samt et tall for maks. antall personer romtypen har plass til, oppgitt ved attributtet ant_personer. Tabellen ETASJE brukes til å vise hvilke etasjer (etasjenummer) i bygninger det finnes rom for reservasjon i – dette kan variere fra bygning til bygning.

Tabellen RESERVASJON inneholder data for reservasjoner av rom. Vi forenkler litt, og lagrer kun epost-adressen til personen som har foretatt reservasjonen i attributtet kontakt_epost. En reservasjon gjelder for en gitt dato og fra- og til klokkeslett, gitt ved attributtene dato, fra_tid, til_tid.

Se for øvrig vedlegg for SQL-script.

Oppgave 3) Programmering – 40%

I denne oppgaven skal du kun jobbe med ett sykehus. Ett sykehusobjekt består av en liste med avdelinger og er identifisert med et navn. Hver avdeling består videre av en liste med ansatte og en liste med pasienter. Ansatte kan videre kategoriseres som leger og sykepleiere.

Vedlegg 2: SQL-script (oppgave 2)

```
CREATE TABLE campus (
  campusid INTEGER PRIMARY KEY,
  camp_navn VARCHAR(30) NOT NULL);

CREATE TABLE bygning (
  bygnnavn VARCHAR(20) PRIMARY KEY,
  campusid INTEGER NOT NULL);

CREATE TABLE etasje (
  bygnnavn VARCHAR(20) NOT NULL,
  etanr INTEGER NOT NULL,
  CONSTRAINT etasje_pk PRIMARY KEY (bygnnavn, etanr));

CREATE TABLE rom (
  romid INTEGER PRIMARY KEY,
 romnavn VARCHAR(30),
  typeid INTEGER NOT NULL,
 bygnnavn VARCHAR(20) NOT NULL,
  etanr INTEGER NOT NULL);

CREATE TABLE romtype (
  typeid INTEGER PRIMARY KEY,
  beskrivelse VARCHAR(200),
  ant_personer INTEGER NOT NULL);

CREATE TABLE reservasjon (
  resid INTEGER PRIMARY KEY AUTO_INCREMENT,
 kontakt_epost VARCHAR(30),
  romid INTEGER NOT NULL,
 dato DATE NOT NULL, -- MySQLs datatype DATE lagrer datoer i formatet 'YYYY-MM-DD'
  fra_tid TIME NOT NULL, -- MySQLs datatype TIME lagrer klokkeslett i formatet 'hh:mm'
  til_tid TIME NOT NULL, -- MySQLs datatype TIME lagrer klokkeslett i formatet 'hh:mm'
  kommentar VARCHAR(300));
```

```
ALTER TABLE bygning
  ADD CONSTRAINT bygning_fk FOREIGN KEY(campusid)
  REFERENCES campus (campusid);

ALTER TABLE etasje
  ADD CONSTRAINT etasje_fk1 FOREIGN KEY(bygnnavn)
 REFERENCES bygning (bygnnavn);

ALTER TABLE rom
  ADD CONSTRAINT rom_fk1 FOREIGN KEY(typeid)
  REFERENCES romtype (typeid);

ALTER TABLE rom
  ADD CONSTRAINT rom_fk2 FOREIGN KEY(bygnnavn,etanr)
 REFERENCES etasje (bygnnavn,etanr);

ALTER TABLE reservasjon
  ADD CONSTRAINT reservasjon_fk1 FOREIGN KEY(romid)
 REFERENCES rom (romid);
```