NTNU
Kunnskap for en bedre verden

FAKULTET FOR INFORMASJONSTEKNOLOGI OG ELEKTROTEKNIKK

Eksamensoppgave i
TDAT1005 Databaser med videregående programmering

Faglig kontakt under eksamen:

Else Lervik, tlf 482 89 200

Anette Wrålsen, tlf 977 96 878

Eksamensdato: 19.mai 2017

Eksamenstid (fra-til): 0900-1400

Hjelpemiddelkode/Tillatte hjelpemidler:

2 håndskrevne A4-ark som studenten selv må ta med seg til eksamen.

Annen informasjon:
Les gjennom hele oppgavesettet før du begynner arbeidet, og disponer tiden.

Dersom noe virker uklart i oppgavesettet, skal du gjøre dine egne antagelser og forklare dette i besvarelsen.

Lykke til!

Målform/språk: Bokmål

Antall sider (uten forside): XX

Antall sider vedlegg: XX

OVERSIKT OVER VEDLEGG

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

Merk: For å få ståkarakter i emnet, må kandidaten ha ståkarakter på Oppgave 1 og 2 hver for seg. B (eller A) forutsetter også ståkarakter på Oppgave 3.

OPPGAVE 1 – DATAMODELLERING OG SQL (45%)

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.

Bedriften Vår jobber med prosjekter for eksterne kunder. I modelleringsoppgaven skal du lage et utdrag av en database som er grunnlaget i et system for prosjektplanlegging, -gjennomføring og –oppfølging. Sentralt står følgende:

- Persondata og ulike kategorier personer
- Prosjektdata med målformuleringer og inndeling i delprosjekt, faser og aktiviteter
- Prosjektorganisasjonen og sammenhengen mellom personer og de enkelte delene av et prosjekt

Videre detaljer:

Et prosjekt har en (ekstern) kunde og en (intern) prosjektleder. Kunden er registrert som et firma i databasen og kan ha mange prosjekter hos oss.

Et prosjekt identifiseres med et nummer. I tillegg lagres et prosjektnavn.

Vi har mange kunder, og en kunde identifiseres med et nummer. I tillegg lagres navn, adresse, epost og tlf.

En person identifiseres med et nummer. I tillegg skal navn, epost og telefon lagres for alle personer. Databasen inneholder flere personkategorier:

- Bedriftens egne ansatte deles inn i kategoriene leder (bl.a. prosjektleder), ingeniør og administrativt ansatt. Om en ansatt lagrer vi timelønn og ansettelsesår.
- Det skal også lagres informasjon om eksterne personer (dvs. personer som ikke er ansatt i Bedriften Vår). Spesielt nevnes kontaktpersoner ansatt hos kundene våre. Disse er ikke knyttet til bestemte prosjekter. Det skal lagres informasjon om rolle og funksjon (en tekst) til hver enkelt av de eksterne personene.

Prosjektets målformuleringer skal lages i databasen. Vi skiller mellom resultatmål og effektmål. Resultatmålene deles inn i delmål. Et mål identifiseres med et nummer, og i tillegg lagres målformuleringen og status som tekster. Status er en tekst som oppdateres underveis av prosjektleder.

Et prosjekt deles inn i flere delprosjekt, som igjen deles inn i faser og videre i aktiviteter.

Delprosjektet identifiseres ved det prosjektet det tilhører sammen med et entydig nummer. Eksempel: Prosjekt 1 deles inn i delprosjektene 1.1 og 1.2, mens prosjekt 2 deles inn i delprosjektene 2.1, 2.2 og 2.3. Om hvert delprosjekt skal budsjett samt planlagt (estimert) og virkelig start- og sluttdato lagres.

De enkelte delprosjektene kan avhenge av hverandre slik at for eksempel delprosjekt C må utføres før delprosjekt D, men kan utføres samtidig med delprosjekt E. Denne avhengigheten må inn i databasen.

Et delprosjekt knyttes til et eller flere delmål. Hvert delmål kan knyttes til flere delprosjekt.

Fasene har et entydig nummer, et budsjett og planlagt og virkelig startdato. Vi trenger ikke å lagre sluttdato, da det alltid er slik at en fase etterfølger den forrige.

En aktivitet tildeles også et entydig nummer. I tillegg lagres en kort beskrivelse og planlagte kostnader (budsjett) for bruk av henholdsvis ingeniører og administrativt ansatte. Vi ser bort fra andre kostnader.

Vi kan ha tilsvarende sammenhenger mellom aktiviteter som mellom delprosjekt.

Til hvert prosjekt knyttes det en prosjektleder som ikke fører timer. Øvrige ansatte (ingeniører og administrativt ansatte) knyttes til et prosjekt ved at de fører timer på aktiviteter som inngår i prosjektet. Antall brukte timer registreres pr uke. (Anta at ukenr er nok, vi ser bort fra årstall.)

I tillegg til prosjektledelse, ingeniører og administrativt ansatte har hvert prosjekt styringskomite, referansegruppe og kvalitetskontrollgruppe. Sammensetningen av disse gruppene skal inn i databasen. Felles for de som deltar i disse gruppene er at de ikke er daglig engasjert i et prosjekt, men de kan være både interne (Bedriften Vår) og eksterne (kunden og andre).

Oppgave a) - Vekt 25%

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

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 b) under er satt opp). Strek under primærnøkler og marker fremmednøkler med stjerne.

Oppgave b) - Vekt 20%

Bedriften Vår tilbyr sine ansatte kurs og sertifiseringer for å øke kompetansen sin, og lagrer informasjon om dette i følgende relasjoner. Attributtene *ans_nr* og *kursholder* er fremmednøkler til relasjonene fra Oppgave a), men er ikke markert i relasjonene under. (Det antas altså at relasjonene fra a) som lagrer informasjon om ansatte identifiserer dem med et nummer.)

KURS(kurs_nr, kurs_navn, beskrivelse, ant_timer, sert_nr*)
KURSKJØRING(kurs_nr*, startdato, kursholder)
DELTAKELSE(ans_nr, (kurs_nr, startdato)*, fullført)
SERTIFISERING(sert_nr, sert_navn)
KOMPETANSE(ans_nr, sert_nr*, dato_oppnådd)

Kursene holdes av erfarne ansatte, og varer et bestemt antall timer. Alle kurs beskrives med en kort tekst. Et fullført kurs kan, men må ikke, gi en sertifisering. Merk at forskjellige kurs kan gi samme sertifisering. Hvis det ikke gir noen sertifisering, settes *sert_nr* til NULL.

Samme kurs kan ha forskjellige kursholder i forskjellige kjøringer. Det er maksimalt en kjøring av et gitt kurs som starter opp på en bestemt dato. En ansatt kan melde seg på samme kurs flere ganger, for eksempel for å friske opp kunnskaper eller fordi han/hun ikke hadde tid til å fullføre forrige gang. Attributten *fullført* er en boolean som holder styr på om den ansatte fullførte den aktuelle kurskjøringen eller ikke.

Bedriften lagrer informasjon om de ansattes sertifiseringer, både sertifiseringer bedriften tilbyr og sertifiseringer fra andre steder (som for eksempel Microsoft-sertifiseringer). De forskjellige sertifiseringene lagres i en egen relasjon. Hvilke ansatte som har hvilke sertifiseringer lagres i relasjonen KOMPETANSE.

Se for øvrig vedlegg for SQL-script.

I resten av oppgaven skal du utelukkende svare ved å skrive SQL-setninger (spørringer) - en eller flere for hver oppgave - for å komme fram til svaret, med utgangspunkt i tabellene over.

Oppgave i)
Skriv ut ansattnummeret til alle ansatte som både har fullført kurs og vært kursholder.

Oppgave ii)
Skriv ut antall forskjellige sertifiseringer som bedriften tilbyr, og antall kurs som ikke tilbyr en sertifisering.

Oppgave iii)
Skriv ut navnet på alle sertifiseringer minst en ansatt har og hvor mange ansatte som har hver av dem, sortert i synkende rekkefølge.

Oppgave iv)
Skriv ut ansattnummeret til eventuelle ansatte som har fullført alle kurs i KURS.

Oppgave v)
Finn navnet på alle sertifiseringer som minst en ansatt har men som bedriften ikke tilbyr.

OPPGAVE 2 – JAVA-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 den delen av systemet som omfatter personinnsats i en prosjektfase. Vi skal se på både planlagt og virkelig timeforbruk. Klassen *Prosjektfase* er gitt:

```
class Prosjektfase {
  private java.util.ArrayList<Aktivitet> aktiviteter = new  java.util.ArrayList<Aktivitet>();
  public double finnRestBudsjett() {  // hvor mye er igjen av budsjettet for denne prosjektfasen?
    // denne metoden skal du programmere etterhvert
  }
}
```

Du skal ikke programmere mot databasen i denne oppgaven.

Oppgave a)

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 i hele Oppgave 2 inkl. klassen *Prosjektfase*.

Oppgave b)

Programmer et klassetre med to typer personkategorier, det vil si klassene *Person*, *Ingenior* og *Adm*.

Konstruktører skal være med, men øvrige metoder begrenses til det du trenger i denne deloppgaven og i resten av programmeringsoppgaven.

Programmer metoden *beregnKostnad()* ved hjelp av polymorfi etter følgende regler:

Kostnadene som faktureres ut til kunden avhenger av timelønn og, for ingeniørene, av ansienniteten.

- Administrativt ansatte faktureres ut med 1,4 * timelønn.

- Ingeniører:

  Ansiennitet 0-2 år: 1,6 * timelønn
  Ansiennitet 3-5 år: 1,8 * timelønn
  Ansiennitet 6 år og mer: 2,1 * timelønn

Ansienniteten er antall år vedkommende har vært ansatt i bedriften. Ansettelsesår skal lagres for alle ansatte, ikke bare ingeniørene. For enkelthets skyld hardkoder du ansienniteten som (2017 – ansettelsesår).

Oppgave c)

Du skal nå programmere klassen *Aktivitet*. Klassen skal inneholde en aktivitetsidentifikasjon (nummer) og variabler for estimerte kostnader for bruk av hhv ingeniører og administrativt ansatte.

Den skal videre ha en metode som registrerer antall timer brukt på denne aktiviteten for en person en gitt uke (se bort fra årstall). Dersom personen er registrert med aktivitet denne uka fra før, skal antall timer plusses på det som allerede er registrert.

Du må selv finne en passende måte å kople aktivitetsobjektet til aktuelle personer på. Se også oppgave d), der du skal bruke dette.

Oppgave d)

Lag en metode i klassen *Aktivitet* for å finne de totale kostnadene for arbeid utført av ingeniører på aktiviteten.

Lag en tilsvarende metode for de administrativt ansatte.

Oppgave e)

Lag metoden *finnRestBudsjett()* vist helt i begynnelsen av Oppgave 2.

Oppgave f)

Lage metoden *hentSortertListeKostnader()* i klassen *Prosjektfase*.

Metoden skal hente ut en liste (tabell) over alle aktiviteter. For hver aktivitet skal listen inneholde aktivitetsidentifikasjon og totale kostnader for arbeid utført av ingeniører og administrativt ansatte.

Listen skal være sortert etter stigende kostnader. Bruk *java.util.Arrays.sort()* til sorteringen.

OPPGAVE 3 – TEORISPØRSMÅL (15%)

Oppgave a)

Gitt følgende kodesnutt:

```
class A {
	public String metode1() {
		return "metode1";
	}
}

class B extends A {
	public String metode1() {
		return super.metode1()+"B";
	}
}

class C extends A {
	public String metode2() {
		return "metode2";
	}
}

class D extends B {
	public String metode1() {
		return super.metode1()+"D";
	}
}

class Test {
	public static void main(String[] args) {
		A objekt1 = new C();
		B objekt2 = new D();
		D objekt3 = new D();
		System.out.println(objekt1.metode1()); // i
		System.out.println(objekt1.metode2()); // ii
		System.out.println(objekt2.metode1()); // iii
		System.out.println(objekt3.metode1()); // iv
	}
}
```

For hver av linjene merket i-iv, avgjør om linja vil gi kompileringsfeil, og hvis ikke, hva utskriften blir.

Oppgave b)

Hendelsesorientert programmering handler blant annet om kommunikasjon mellom *lyttere* og *kilder*. Forklar hvordan dette foregår i Java, med utgangspunkt i en bruker som klikker på en knapp i et vindu på skjermen.

Oppgave c)

Det finnes tre typer problemer som kan oppstå ved en transaksjonsutføring, nemlig tapt oppdatering, ikke-overgitte data (kalles noen ganger ikke-bekreftede data) og inkonsistente innhentinger (kalles noen ganger inkonsistent uthenting). Forklar kort hva hvert av disse problemene går ut på, illustrert ved et enkelt eksempel.

VEDLEGG 1: CASE-BESKRIVELSEN, SLIK DENNE LÅ I ITSLEARNING

Bakgrunnsstoff

Leksjonen «Prinsipper for prosjektarbeid» fra TDAT1007 Ingeniørfaglig innføringsemne. Les gjennom denne på nytt og forsikre deg om at du er fortrolig med språkbruken. Lenke: http://iie.ntnu.no/fag/_innfIng/else/Prinsipper-prosjektarbeid-leksjon.pdf (Lenken er ikke tilgjengelig på eksamensdagen.)

OBS: Leksjonen bruker begrepene «hovedoppgave» og «delprosjekt». I vår sammenheng er dette synonymer, og i eksamenssettet blir kun begrepet delprosjekt brukt:

Et *prosjekt* deles inn i *delprosjekt* som igjen deles inn i *faser* og videre i *aktiviteter*.

Oppgave 1a) Datamodellering – 25%

Bedriften Vår jobber med prosjekter for eksterne kunder. I modelleringsoppgaven skal du lage et utdrag av en database som er grunnlaget i et system for prosjektplanlegging, -gjennomføring og –oppfølging. Sentralt står følgende:

- Persondata og ulike kategorier personer
- Prosjektdata med målformuleringer og inndeling i delprosjekt, faser og aktiviteter
- Prosjektorganisasjonen og sammenhengen mellom personer og de enkelte delene av et prosjekt

Oppgave 1b) SQL – 20%

Bedriften Vår tilbyr sine ansatte kurs og sertifiseringer for å øke kompetansen sin, og lagrer informasjon om dette i følgende relasjoner. Attributtene *ans_nr* og *kursholder* er fremmednøkler til relasjonene fra Oppgave a), men er ikke markert i relasjonene under. (Det antas altså at relasjonene fra a) som lagrer informasjon om ansatte identifiserer dem med et nummer.)

KURS(kurs_nr, kurs_navn, beskrivelse, ant_timer, sert_nr*)
KURSKJØRING(kurs_nr*, startdato, kursholder)
DELTAKELSE(ans_nr, (kurs_nr, startdato)*, fullført)
SERTIFISERING(sert_nr, sert_navn)
KOMPETANSE(ans_nr, sert_nr*, dato_oppnådd)

Kursene holdes av erfarne ansatte, og varer et bestemt antall timer. Alle kurs beskrives med en kort tekst. Et fullført kurs kan, men må ikke, gi en sertifisering. Merk at forskjellige kurs kan gi samme sertifisering. Hvis det ikke gir noen sertifisering, settes sert_nr til NULL.

Samme kurs kan ha forskjellige kursholder i forskjellige kjøringer. Det er maksimalt en kjøring av et gitt kurs som starter opp på en bestemt dato. En ansatt kan melde seg på samme kurs flere ganger, for eksempel for å friske opp kunnskaper eller fordi han/hun ikke hadde tid til å fullføre forrige gang. Attributten *fullført* er en boolean som holder styr på om den ansatte fullførte den aktuelle kurskjøringen eller ikke.

Bedriften lagrer informasjon om de ansattes sertifiseringer, både sertifiseringer bedriften tilbyr og sertifiseringer fra andre steder (som for eksempel Microsoft-sertifiseringer). De forskjellige

sertifiseringene lagres i en egen relasjon. Hvilke ansatte som har hvilke sertifiseringer lagres i relasjonen KOMPETANSE.

Se for øvrig vedlegg for SQL-script.

Oppgave 2 Programmering – 40%

I programmeringsoppgaven skal du jobbe med den delen av systemet som omfatter personinnsats i en prosjektfase. Vi skal se på både planlagt og virkelig timeforbruk. Klassen *Prosjektfase* er gitt:

```
class Prosjektfase {
  private java.util.ArrayList<Aktivitet> aktiviteter = new  java.util.ArrayList<Aktivitet>();
  public double finnRestBudsjett() {  // hvor mye er igjen av budsjettet for denne prosjektfasen?
     ……
  }
}
```

VEDLEGG 2: SQL-SCRIPT (OPPGAVE 1)

```
CREATE TABLE KURS(
  kurs_nr INTEGER NOT NULL AUTO_INCREMENT,
  kurs_navn VARCHAR(30) NOT NULL,
  beskrivelse TEXT,
  ant_timer INTEGER NOT NULL,
  sert_nr INTEGER,
  CONSTRAINT kurs_pk PRIMARY KEY (kurs_nr));

CREATE TABLE KURSKJØRING(
  kurs_nr INTEGER NOT NULL,
  startdato DATE NOT NULL,
  kursholder INTEGER,
  CONSTRAINT kurskjøring_pk PRIMARY KEY (kurs_nr,startdato));

CREATE TABLE DELTAKELSE(
  ans_nr INTEGER NOT NULL,
  kurs_nr INTEGER NOT NULL,
  startdato DATE NOT NULL,
  fullført BOOLEAN,
  CONSTRAINT deltakelse_pk PRIMARY KEY (ans_nr, kurs_nr, startdato));

CREATE TABLE SERTIFISERING(
  sert_nr INTEGER NOT NULL AUTO_INCREMENT,
  sert_navn VARCHAR(30) NOT NULL,
  CONSTRAINT sertifisering_pk PRIMARY KEY (sert_nr));

CREATE TABLE KOMPETANSE(
  ans_nr INTEGER NOT NULL,
  sert_nr INTEGER NOT NULL,
  dato_oppnådd DATE NOT NULL,
  CONSTRAINT kompetanse_pk PRIMARY KEY (ans_nr, sert_nr));

--- Fremmednøkler
ALTER TABLE KURS
  ADD CONSTRAINT kurs_fk FOREIGN KEY (sert_nr)
  REFERENCES SERTIFISERING(sert_nr);

ALTER TABLE KURSKJØRING
```

```
  ADD CONSTRAINT kurskjøring_fk FOREIGN KEY (kurs_nr)
  REFERENCES KURS(kurs_nr);

ALTER TABLE DELTAKELSE
  ADD CONSTRAINT deltakelse_fk FOREIGN KEY (kurs_nr,startdato)
  REFERENCES KURSKJØRING(kurs_nr, startdato);

ALTER TABLE KOMPETANSE
  ADD CONSTRAINT kompetanse_fk FOREIGN KEY (sert_nr)
  REFERENCES SERTIFISERING(sert_nr);
```