NTNU
Kunnskap for en bedre verden

Fakultet for teknologi

Eksamensoppgave i
TDAT1005 Databaser med videregående programmering

Faglig kontakt under eksamen:

Else Lervik, tlf 73 55 95 50

Eksamensdato: 09.mai 2016

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): 8

Antall sider vedlegg: 2

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)
3. Klassen Kunde til bruk i oppgave 2.

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

OPPGAVE 1 – DATAMODELLERING OG SQL (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.

I modelleringsoppgaven skal du arbeide med følgende elementer i mobiloperatørens database:

- Kundeoversikt
- Abonnementstyper
- Abonnementer med fakturering
- Tilleggstjenester
- Samtaler, både til spesialnummer og til utlandet
- Nettbutikk med salg av telefoner med tilbehør

Følgende gjelder:

Mobilselskapet tilbyr ulike typer abonnementer, se eksempler, tabell oppgave 2. Det skal tilrettelegges for enkel endring og utvidelse av abonnementstilbudet basert på månedspris og fri-grenser for MMS, SMS og datatrafikk (antall GB). Samtale til norske mobil- og fasttelefon-nummer er gratis.

Tilleggstjenester i form av tjenester med en engangspris (eksempler: gullnummer og sølvnummer) og tjenester med månedspris (eksempler: spesifisert regning og telefonforsikring) skal inn i databasen.

Prisliste for spesialnummer og utenlandssamtaler skal registreres med en startavgift og en ringepris pr minutt. Den enkelte samtalen knyttes til det aktuelle abonnementet, og startidspunkt og varighet registreres.

For øvrig aktivitet vedlikeholdes tellere for antall SMS, antall MMS og antall Gigabyte pr måned. Når måneden er slutt, kjøres et program som genererer fakturaer. Fakturaene omfatter all aktivitet knyttet til abonnementet den siste måneden. Samtidig slettes all detaljinformasjon om aktiviteten fra databasen. Disse detaljene finner man nå i fakturaen, det vil si at fakturaene som sådanne (f.eks. som navn på pdf-filer), lagres i databasen. En faktura identifiseres med tlfnr, måned og år og har en betalingsfrist. Betaling og eventuell purring (med ny frist) skal registreres i databasen.

Mobilselskapet tilbyr nettbutikk der de selger telefoner med tilbehør. Hver vare identifiseres med et entydig varenummer. I tillegg skal navn, modell, beskrivelse, pris og lagerstatus (antall på lager) registreres. Sammenheng mellom telefon og tilbehør skal framgå. Samme type tilbehør kan brukes på flere ulike telefonmodeller.

En ordre identifiseres med et nummer. Ordren knyttes til en bestemt kunde (ikke til et abonnement), og opplysninger om ordredato, betalingsmåte, eventuell betalingsfrist og ordrestatus skal lagres i databasen. I en og samme ordre kan man bestille både telefoner og tilbehør, uavhengig av hverandre. Det må være mulig å angi et antall av den varen man bestiller. En kunde identifiseres med en entydig kundeid.

Oppgave a) - Vekt 15%

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

> Merk at det i enkelte tilfeller kan være fornuftig å bruke 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 b) under er satt opp). Strek under primærnøkler og marker fremmednøkler med stjerne.

Oppgave b) - Vekt 5%

Vi ønsker å utvide databasen i Oppgave a) med følgende relasjoner (tabeller) skrevet på relasjonell form med primær- og fremmednøkler (fremmednøkler som refererer til Oppgave a) er ikke markert). Tabellene inneholder data om kundevurderinger av varene i nettbutikken.

> VURDERING(vurd_id, kundeid, varenr, omtale, anbefaling)
> VURD_SCORE(vurd_id*, krit_id*, score)
> VURD_KRITERIUM(krit_id, navn, maks_score)
> VARE_KRITERIUM(varenr, krit_id*)

En vurdering er gitt av en registrert kunde, identifisert med attributtet kundeid, for en gitt vare, identifisert med attributtet varenr (varenummer). En vurdering inneholder en tekstlig omtale og en (kjøps)anbefaling, med verdi ja eller nei (boolean). I tillegg gir kunden score (poeng) til varen ut fra gitte kriteria. Ulike vurderingskriteria er gitt i tabellen VURD_KRITERIUM. Maksimum score for et vurderingskriterium er gitt i attributtet maks_score. Minimum score kan vi si er konstant lik 1.0 for alle vurderingskriteria. Vi forutsetter at en kunde må oppgi en score for alle kriteria for en vare (for at det bl.a. skal være enkelt å regne ut gjennomsittsscore for en vare), og at enhver registrert verdi i attributtet score ikke er høyere enn aktuell maks_score.

Se for øvrig vedlegg for SQL-script.

OPPGAVE:

Utvid ER/EER-modellen i Oppgave a) til å inkludere entitetstyper og sammenhengstyper for tabellene over.

Merk deg: Du trenger ikke å tegne ER/EER-modellen i Oppgave a) på nytt. Du kan tegne utvidelsen på et eget ark så lengde det kommer tydelig frem hvor du vil plassere de nye entitets- og sammenhengstypene i modellen.

SQL-oppgaver

I resten av Oppgave 1 skal du utelukkende skrive SQL-setninger (spørringer) - en eller flere for hver oppgave - for å komme fram til svaret.

Oppgave c)

Bruk tabellene i Oppgave b) til å svare på følgende:

Skriv ut varenummer og antall vurderinger til alle varer som minst tre kunder har vurdert, sortert slik at varer med flest vurderinger skrives ut først i resultatet.

Oppgave d)

Bruk tabellene i Oppgave b) til å svare på følgende:

Skriv ut varenummer til de varene som minst en kunde har gitt maksimum score på samtlige (alle) vurderingsscorene til varen.

Oppgave e)

Bruk tabellene i Oppgave b) til å svare på følgende:

Skriv ut varenummer, evt. varenumrene hvis det er flere varer, til varen/varene med best (høyest) gjennomsnittsscore for vurderingskriteria med maks_score = 5.0.

Oppgave f)

I denne oppgaven skal du bruke tabeller fra **både** Oppgave a) og b) til å svare på følgende:

Skriv ut kundeid og kundenavn til alle kunder som har vurdert minst en av varene som de også har lagt inn en ordre på (d.v.s. kunden har vurdert en vare som han/hun har kjøpt).

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.

Programmeringsoppgaven omfatter mobilselskapets kunde- og abonnement-register. Kundene kan velge mellom flere ulike abonnementstyper – dog ikke så generelt som oppgave 1.

Dersom samme kunde betaler for flere abonnementer, innvilges familie-/vennerabatt for alle med samme betaler.

Vi skal i denne oppgaven bruke et forenklet sett av abonnementstyper, der vi ser bort fra f.eks. samtaler med utlandet. Tale og SMS er gratis for alle abonnementstyper.

Følgende abonnementstyper gjelder i denne oppgaven:

Navn	Månedspris	Antall MMS gratis	Pris pr MMS utover grensen	Antall GB gratis	Pris pr GB utover grensen
Standard	99,-	100	2,-	0	79,-
DataPluss	249,-	100	2,-	3	29,-
DataPlussPluss	379,-	100	2,-	10	29,-
GratisMMS	199,-	Ubegrenset	0	0	79,-

Vennerabatten er på 30 kr/abonnement og trer i kraft fra og med at en kunde er registrert som betaler av minst to abonnementer.

Kunde- og abonnement-registrene skal implementeres i klassen *Mobilselskap* som følger:

```
class Mobilselskap {
  private java.util.ArrayList<Abonnement> abonnementer = new java.util.ArrayList<Abonnement>();
  private java.util.ArrayList<Kunde> kunder = new java.util.ArrayList<Kunde>();
  …
}
```

Klassen *Kunde* er gitt i Vedlegg 3, men kan utvides/endres dersom du mener at det er hensiktsmessig.

Klassen *Abonnement* skal ha en referanse til kunden som betaler, samt objektvariabler for forbruket (tale i minutter, antall SMS, antall MMS, antall GB data) siste måned. Et objekt av klassen må også ha informasjon om telefonnummeret og eventuell rabatt knyttet til abonnementet. Vi antar at når måneden er slutt, lages faktura (som eksporteres, du ser bort fra dette i denne oppgaven) og forbruket nullstilles.

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. klassene *Mobilselskap* og *Kunde*.

Oppgave b)

Programmer et klassetre av de ulike abonnementstypene.

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

Programmer metoden *beregnBeløp()* ved hjelp av polymorfi slik at fakturabeløpet regnes ut på grunnlag av opplysningene foran.

Oppgave c)

indexOf() er en metode i klassen *java.util.ArrayList*. Eksempel på bruk:

```
int pos = kunder.indexOf(enKunde);  // enKunde er et objekt av klassen Kunde
```

Her vil *pos* være lik indeksen til første forekomst av et objekt likt med *enKunde*. Hvis et slikt objekt ikke eksisterer, har *pos* fått en negativ verdi.

kundeId identifiserer en kunde entydig, og vi kan derfor anta at to kundeobjekter er like dersom *kundeId* er den samme i begge objektene.

- Programmer det som er nødvendig for at *indexOf()* skal fungere som beskrevet for arraylisten *kunder*.
- Hva skjer dersom du ikke gjør dette?

Du kan i det følgende anta at *indexOf()* kan brukes mot begge listene (*kunder* og *abonnementer*) i oppgaven. Abonnementer identifiseres som like hvis de har samme telefonnummer.

Oppgave d)

Du skal nå lage en metode i klassen *Mobilselskap*. Metoden skal brukes til å registrere et nytt abonnement av riktig type, og den skal ha følgende hode:

public Abonnement regNyAb(Kunde betaler, String abType) // adType er et klassenavn

Kunden skal registreres, dersom han/hun ikke allerede fins i registeret.

Du kan anta at det eksisterer en metode *finnLedigTlfnr()* som gir deg et telefonnummer som kan brukes.

Eventuell vennerabatt skal legges inn. Dersom dette er abonnement nr to for kunden, skal rabatten også legges inn i det første abonnementet som kunden har.

Metoden skal returnere *null* dersom *abType* ikke er en gyldig abonnementstype, ellers returneres en referanse til det nye objektet.

Oppgave e)

Lag en metode som endrer abonnementstype for et gitt telefonnummer. Metoden skal ha følgende hode:

```
public Abonnement endreAb(Abonnement ab, String tilType)
```

tilType skal svare til et klassenavn. Dersom klassenavnet er ugyldig, skal metoden returnere *null*, hvis ikke, skal metoden returnere en referanse til det endrede abonnementet.

OPPGAVE 3 – TEORISPØRSMÅL (20%)

Oppgave a)

Hva er et interface i Java?

Eksempler på interface i Java-API-et er *Comparator*, *Comparable* og *ActionListener*. Hva brukes disse tre interfacene til, og hvorfor er det hensiktsmessig å bruke interface her?

Oppgave b)

For å lage Java-program mot en database, trenger vi en JDBC-databasedriver. Hva er en JDBC-databasedriver? Hva består den av, og hvorfor er den bygd opp slik?

Oppgave c)

En studentforening tjener noen kroner på å distribuere selvlagede kompendier tilpasset fag på universitetet de er tilknyttet. Oversikten over alle kompendier de tilbyr er en implementasjon av følgende relasjon:

(Fag, Årstall, Tittel, Format, Pris)

Det lages høyst et kompendium per fag per år, og hvert kompendium kommer kun i et format (bok, hefte eller elektronisk). Alle kompendier i samme format koster det samme, uansett fag.

(i) Beskriv alle (fulle) funksjonelle avhengigheter i relasjonen over.
(ii) Forklar hvorfor relasjonen er i 2NF.
(iii) Forklar hvorfor relasjonen ikke er i 3NF.

Oppgave d)

Forklart kort forskjellen på en leselås og en skrivelås i transaksjonshåndtering, og hvordan slike låser kan brukes til å oppnå en serialiserbar gjennomføring av samtidige transaksjoner.

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

Oppgave 1 og 2 (Datamodellering og programmering):

Du skal jobbe med database og programsystem for en mobiloperatør.

I modelleringsoppgaven skal du jobbe med kunder, abonnementer, abonnementstyper, nettbutikken, fakturering og ulike tilleggstjenester.

I programmeringsoppgaven skal du begrense deg til kunder og noen bestemte abonnementstyper med fakturering. Utgangspunktet skal være klassen *Mobilselskap* med følgende lister:

```
class Mobilselskap {
  private java.util.ArrayList<Abonnement> abonnementer = new java.util.ArrayList<Abonnement>();
  private java.util.ArrayList<Kunde> kunder = new java.util.ArrayList<Kunde>();
   …
}
```

Oppgave 1 (SQL-del):

Følgende relasjoner (tabeller) er skrevet på relasjonell form med primær- og fremmednøkler. Tabellene inneholder data om kundevurderinger av varene i nettbutikken.

VURDERING(vurd_id, kundeid, varenr, omtale, anbefaling)
VURD_SCORE(vurd_id*, krit_id*, score)
VURD_KRITERIUM(krit_id, navn, maks_score)
VARE_KRITERIUM(varenr, krit_id*)

En vurdering er gitt av en registrert kunde, identifisert med attributtet kundeid, for en gitt vare, identifisert med attributtet varenr (varenummer). En vurdering inneholder en tekstlig omtale og en (kjøps)anbefaling, med verdi ja eller nei (boolean). I tillegg gir kunden score (poeng) til varen ut fra gitte kriteria. Ulike vurderingskriteria er gitt i tabellen VURD_KRITERIUM. Maksimum score for et vurderingskriterium er gitt i attributtet maks_score. Minimum score kan vi si er konstant lik 1.0 for alle vurderingskriteria. Vi forutsetter at en kunde må oppgi en score for alle kriteria for en vare (for at det bl.a. skal være enkelt å regne ut gjennomsittsscore for en vare), og at enhver registrert verdi i attributtet score ikke er høyere en aktuell maks_score.

Se for øvrig vedlegg for SQL-script.

VEDLEGG 2: SQL-SCRIPT (OPPGAVE 1)

```
CREATE TABLE VURDERING(
 vurd_id INTEGER,
 kundeid INTEGER NOT NULL,
 varenr INTEGER NOT NULL,
 omtale TEXT,
 anbefaling BOOLEAN,
 CONSTRAINT pk_vurdering PRIMARY KEY(vurd_id));
```

```
CREATE TABLE VURD_KRITERIUM(
 krit_id INTEGER,
 navn CHAR(30) NOT NULL,
 maks_score DECIMAL(3,1) NOT NULL,
 CONSTRAINT pk_krit PRIMARY KEY(krit_id));

CREATE TABLE VARE_KRITERIUM(
 varenr INTEGER,
 krit_id INTEGER,
 CONSTRAINT pk_varekrit PRIMARY KEY(varenr, krit_id));

CREATE TABLE VURD_SCORE(
 vurd_id INTEGER,
 krit_id INTEGER,
 score DECIMAL(3,1) NOT NULL,
 CONSTRAINT pk_vurdscore PRIMARY KEY(vurd_id, krit_id));

ALTER TABLE VARE_KRITERIUM
 ADD CONSTRAINT krit_fk1 FOREIGN KEY(krit_id)
 REFERENCES VURD_KRITERIUM (krit_id);

ALTER TABLE VURD_SCORE
 ADD CONSTRAINT vurdscore_fk1 FOREIGN KEY(vurd_id)
 REFERENCES VURDERING (vurd_id);

ALTER TABLE VURD_SCORE
 ADD CONSTRAINT vurdscore_fk2 FOREIGN KEY(krit_id)
 REFERENCES VURD_KRITERIUM (krit_id);
```

VEDLEGG 3: KLASSEN KUNDE (OPPGAVE 2)

```
class Kunde {
  private final int kundeId;
  private final String navn;
  private final String epost;

  public Kunde(int kundeId, String navn, String epost) {
    this.kundeId = kundeId;
    this.navn = navn;
    this.epost = epost;
  }

  public int getKundeId() {
    return kundeId;
  }

  public String getNavn() {
    return navn;
  }

  public String getEpost() {
    return epost;
  }
}
```