UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i :	IN 212 — Databaseteori
Eksamensdag :	Torsdag 8. juni 1995
Tid for eksamen :	09.00 - 15.00
Oppgavesettet er på :	5 sider
Vedlegg :	Ingen
Tillatte hjelpemidler :	Alle trykte og skrevne

Kontroller at oppgavesettet er komplett før du begynner å besvare det

Oppgave 1 (35 %)

Nedenfor følger (deler av) det begrepsmessige skjema for en database for registrering av eksamener og eksamensresultater:

STUDENT	(SNum, Navn, Adr, SumVekttall)
OPPMELDING	(SNum, KursNum, Semester)
EKSAMEN	(KursNum, SNum, Semester, Karakter)
KURS	(KursNum, KursNavn, Vekttall)

Vekttall og SumVekttall er heltall. Karakter er desimaltall med én desimal. Alle andre felt er tekstfelt. Nil-verdier er ikke tillatt i noe felt.

SNum er primærnøkkel i STUDENT
(SNum, KursNum, Semester) er primærnøkkel i OPPMELDING
(KursNum, SNum, Semester) er primærnøkkel i EKSAMEN
KursNum er primærnøkkel i KURS

KursNum i OPPMELDING og EKSAMEN er fremmednøkler til KURS
SNum i OPPMELDING og EKSAMEN er fremmednøkler til STUDENT

Semestrene angis som 'h91', 'v92', 'h92' osv.

Karakterene angis som 1.0, 1.1, ... , 6.0, 7.7 (hvor 7.7 betyr „ikke møtt” eller „trukket seg under eksamen”). 1.0 er beste og 4.0 dårligste ståkarakter.

Oppgave 1 fortsetter på neste side

(Oppgave 1 forts. fra side 1)

Ved emnepåmeldingen legges data i OPPMELDING. Når karakterene blir registrert etter eksamen, sletter programmet dataene i OPPMELDING og flytter dem til EKSAMEN.

Studenter skal bare ha vekttall for bestått kurs, og de kan ikke få vekttall for samme kurs mer enn én gang. For studenter som har stått flere ganger i samme kurs, er det beste karakter som gjelder.

Oppgave 1 A Bruk SQL for å finne navn og adresse til de studentene som stod til eksamen i 'in105' dette semesteret (v95).

Oppgave 1 B Besvar oppgave 1 A i relasjonsalgebra og tuppelkalkyle.

Oppgave 1 C Skriv en UPDATE setning i SQL som setter riktig antall vekttall i attributtet SumVekttall i STUDENT.

Oppgave 1 D Bruk SQL for å lage en liste over navn, sum vektall og gjennomsnittskarakter for hver av de studentene som har bestått (i hvert fall) kursene 'in105', 'in110' og 'in212'. Listen skal være sortert på gjennomsnittskarakter med de beste studentene først.

Oppgave 1 E Lag et konservativt nettverksdatabaseskjema for den gitte datastrukturen.

(Det at skjemaet er konservativt, betyr at ingen informasjon er lagret mer enn ett sted. F.eks. lagres KursNum bare i KURS og verken i EKSAMEN eller OPPMELDING.)

Oppgave 1 F Skriv (et utsnitt av) en skisse av et program (i fritt valgt pseudokode) som brukes til å registrere karakterer etter en eksamen. Gå ut fra at kursnummer og semester er kjent. For hver oppmelding som gjelder denne eksamenen, skal studentnummeret skrives ut på skjermen, hvoretter brukeren skriver inn karakteren. Svaret '0.0' betyr at oppmeldingen ikke skal registreres i EKSAMEN.

Programmet skal bruke datastrukturen fra oppgave 1 E, og det skal slette poster i OPPMELDING og opprette nye i EKSAMEN. Legg vekt på å få frem programstrukturen og hvordan du bruker databasen.

Du kan bruke ACCEPT(X) for å lese en verdi fra tastaturet inn i programvariabelen X og DISPLAY(X) for å skrive verdien av X til skjermen.

Oppgave 2 (15 %)

La R være et relasjonsskjema og la X, Y og Z være attributter i R.

(Merk at vi ikke har forutsatt at X, Y og Z er disjunkte, og heller ikke at de er en dekomposisjon av R.)

Bevis (eller motbevis) følgende slutningsregel for MVD-er:

$$(X \twoheadrightarrow Y \wedge X \twoheadrightarrow Z) \Rightarrow X \twoheadrightarrow YZ$$

der YZ (som vanlig) er en kompakt notasjon for $Y \cup Z$.

Oppgave 3 (15 %)

La R være en relasjon med skjema **R**(R) = {A,B,C,D,E,F,G,H}.

La $\mathcal{F}$ = {A $\rightarrow$ GH, AB $\rightarrow$ F, CD $\rightarrow$ ABE, H $\rightarrow$ C, DEG $\rightarrow$ A} være gitt, og la

$\mathcal{D}$ = {ABC, ADEG, CDF, DEGH} være en dekomposisjon av **R**(R).

Oppgave 3 A Vis at $\mathcal{D}$ ikke er tapsfri.

Oppgave 3 B Vis at vi kan gjøre $\mathcal{D}$ tapsfri ved å bytte ut én av de åtte FD-ene i $\mathcal{F}$ med en annen FD med atomær høyreside.

(Det er flere riktige svar på denne oppgaven, men du trenger bare finne ett.)

Oppgave 4 (15 %)

La R være en relasjon med intensjon {A,B,C,D}.

La $\mathcal{F}$ = {A $\rightarrow$ D, B $\rightarrow$ C, CD $\rightarrow$ B} være gitt.

Oppgave 4 A Hvilke kandidatnøkler har R? Begrunn svaret.

Vis at R ikke er på 5NF.

Finn den høyeste normalform R tilfredsstiller.

Oppgave 4 B Finn alle rene, tapsfrie dekomposisjoner av R til 5NF og avgjør om noen av dem er FD-bevarende.

Utvid $\mathcal{F}$ med MVD-en A —» B, dvs at $\mathcal{F}$ = {A $\rightarrow$ D, B $\rightarrow$ C, DC $\rightarrow$ B, A —» B}

Oppgave 4 C Hvilke kandidatnøkler har R nå, og hva er den høyeste normalform R tilfredsstiller?

Oppgave 5 (20 %)

I denne oppgaven skal vi se på problemet med låsing i transaksjoner i et flerbrukersystem. En minimal relasjonsdatabase for å håndtere bestilling av flybilletter kan se slik ut:

RUTE	(rutenr, fra, til)
FLYAVGANG	(rutenr, dato, antall_seter, ledige_seter)
BESTILLING	(rutenr, dato, passasjernavn)

Vi ser at rutenr er primærnøkkel i RUTE, at (rutenr, dato) er primærnøkkel i FLYAVGANG, og at hele skjemaet er primærnøkkel i BESTILLING.

Vi skal anta at brukerne av systemet bare har følgende to transaksjonstyper til disposisjon (de kan altså bare bestille og avbestille én billett om gangen) :

Type 1: Bestilling:

```
begin_transaction
Les FLYAVGANG;
Hvis ledige_seter > 0 så
      begin
      Skriv ny BESTILLING;
      Mink ledige_seter med 1;
      Oppdater FLYAVGANG
      end
end_transaction;
```

Type 2: Avbestilling:

```
begin_transaction
Fjern BESTILLING;
Les FLYAVGANG;
Øk ledige_seter med 1;
Oppdater FLYAVGANG
end_transaction;
```

Oppgave 5 fortsetter på neste side

(Oppgave 5 forts. fra side 4)

Oppgave 5 A Forutsett først at vi ikke bruker låser. Beskriv en situasjon hvor vi kan få inkonsistens i data hvis en bruker bestiller en billett samtidig som en annen bruker avbestiller på samme flyavgang.

Oppgave 5 B Vi tar så i bruk låser. Drøft kort (høyst én side) på hvilket nivå du vil låse data: På database-, tabell- eller tuppel-nivå. Forklar, både for bestillinger og avbestillinger, hvilke deler av databasen du vil låse, og når du vil låse dem i transaksjonen.

Oppgave 5 C Anta så at vi bruker låser i hver transaksjon. Vis at vi kan få et problem hvis to brukere samtidig prøver å bestille:

– en tur/retur-billett Oslo-Tromsø-Oslo
– en tur/retur-billett Tromsø-Oslo-Tromsø

og disse blir bestilt i den rekkefølgen reisene skal foretas.

Kan denne situasjonen gi vranglås (deadlock), dvs. at begge brukerne venter på hverandre? Begrunn svaret.

Oppgave 5 D Kan vi få problemer i eksemplet i oppgave 5 C hvis den siste brukeren bestiller de to enkeltreisene i omvendt rekkefølge (dvs først tilbakereisen fra Oslo til Tromsø og deretter fremreisen fra Tromsø til Oslo)?

Kan du generalisere svaret ditt til en strategi for bestilling av tur/retur-billetter? (Husk at vi har forutsatt at vi bare kan låse én bestilling om gangen.)

Slutt på oppgavesettet

Arne Maus *Ragnar Normann*