

Andre sett obligatoriske oppgaver i INF3100 V2008

Opgavesettet skal i utgangspunktet løses av grupper på to og to studenter som leverer felles besvarelse. Vi godkjenner også individuelle besvarelser, men oppfordrer dere altså til heller å finne en å samarbeide med. Vi godkjenner ikke grupper på mer enn to studenter. Vi vil foreta stikkprøver der vi plukker ut enkeltpersoner som må gjennomgå sin besvarelse med oss.

Gjennomføring og innlevering av oppgaven skal skje i henhold til gjeldende retningslinjer ved Institutt for informatikk, se

<http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf> (norsk)

<http://www.ifi.uio.no/studinf/skjemaer/declaration.pdf> (engelsk)

*Enhver innlevering av besvarelse på en obligatorisk oppgave tas som en bekref-
telse på at retningslinjene er lest og forstått.*

Innleveringsfrist: Fredag 9. mai kl. 12:00.

Fristen er absolutt, og det blir ikke gitt utsettelse. Alle spørsmålene må besvares for å få godkjent besvarelsen.

Skriv fullt navn og brukernavn på besvarelsen. Besvarelsen sendes med e-post på PDF-format til gruppelærer. E-posten med besvarelsen skal ha følgende subjectfelt:

Subject: Oblig 2 inf3100 (<brukernavn student>)

Studenter som har fått godkjent den obligatoriske oppgaven og likevel vil trekke seg fra eksamen, må levere en papirkopi til gruppelæreren for å få en påtegning om at oppgaven er godkjent. Dette gjelder bare studenter som trekker seg før 14-dagersfristen.

Vi skal se på en enkel relasjonsdatabase for en verktøygrossist. Kundene av grossisten er detaljister (butikker eller foretak, gjerne også enkeltpersoner, som videreselger verktøy til privatpersoner og profesjonelle). Detaljistene kan bestille varer fra grossisten via en internettapplikasjon. Grossisten har flere varelagre rundt om i Norge; når en kunde bestiller varer, plukkes varene fra

det eller de lagrene som ligger nærmest kundens leveringsadresse.

Databasen inneholder en relasjon *Ansatt* hvor hver ansatt er tilordnet et entydig ansattnummer og hvor man kan finne de ansattes navn og adresse. Videre inneholder databasen informasjon om varer og varelagrene til grossisten. Relasjonen *Vare* angir for hvert vareslag en tilhørende entydig kode, et navn på varen og enhetspris. Relasjonen *Lagerbeholdning* angir hvor mange enheter av en vare som befinner seg på de enkelte lagrene.

Varene er klassifisert i varegrupper, denne klassifikasjonen fins i *Varegruppetilhørighet*. En vare kan tilhøre flere varegrupper. Endel ansatte har spesialkunnskap om visse varegrupper. Hvem som har slik kunnskap, fremgår av relasjonen *Ansvarsområde*.

I relasjonen *Kunde* er det for hver kunde registrert et entydig kundenummer, kundens navn, om det er et enpersonsforetak, adresse og foretaksnummer. For å holde rede på bestillingene har databasen to relasjoner *Ordrelinje* og *Ordre*. Hver ordre har et ordrenummer *ordreID*. I relasjonen *Ordrelinje* er det til hver vare i en ordre angitt hvor mange enheter kunden har bestilt av varen. I *Ordre* finner man opplysninger om hvilken kunde ordren gjelder og datoen da bestillingen ble registrert.

I databaseskjemaet har primærnøkler to understrekninger. Andre kandidatnøkler har én understrekning. Det er fremmednøkler fra *Lagerbeholdning* og *Varegruppetilhørighet* til *Vare*, fra *Ansvarsområde* til *Ansatt*, fra *Ordrelinje* til *Vare* og *Ordre*, og fra *Ordre* til *Kunde*. Hvilke attributter som utgjør fremmednøkler, fremgår ved at det er valgt samme attributtnavn i fremmednøkkel som i den relasjonen fremmednøkkelen refererer.

Ansatt (*ansattID*, *navn*, *adresse*)
Kunde (*kundeID*, *navn*, *ftype*, *adresse*, *foretaksnr*)
Vare (*vareID*, *varenavn*, *enhpris*)
Lagerbeholdning (*vareID*, *lagernavn*, *antall*)
Varegruppetilhørighet (*vareID*, *varegr*)
Ansvarsområde (*varegr*, *ansattID*)
Ordrelinje (*ordreID*, *vareID*, *antbest*)
Ordre (*ordreID*, *kundeID*, *dato*)

(1)

Attributtet *dato* i *Ordre* antas å være en standard SQL DATE, dvs. tekst i f.eks. formatet '2007-12-31'. *Kunde* kan være en enkeltperson eller en bedrift,

dette fremgår av attributtet *ftype* i *Kunde* som for enpersonsforetak er 'E', mens bedrifter har 'B'. Hvis vedkommende ikke er en bedrift, vil *foretaksnr* være personnummer istedet.

1 FDer og MVDer

Betrakt følgende alternativ til databaseskjemaet (1) over:

Ansatt (*ansattID*, *navn*, *adresse*)
Vare (*vareID*, *varenavn*, *enhpris*)
Lagerbeholdning (*vareID*, *lagernavn*, *antall*)
Varegruppeinfo (*vareID*, *varegr*, *ansattID*)
Ordrelinje (*ordreID*, *vareID*, *antbest*)
Ordreinfo (*ordreID*, *dato*, *kundeID*, *navn*, *ftype*, *adresse*, *foretaksnr*)

(2)

- (i) Lovlige instanser av *Ordreinfo* i skjema (2) skal reflektere nøyaktig de lovlige instansene til relasjonene *Kunde* og *Ordre* i skjema (1). Hvilke FDer må i såfall gjelde i *Ordreinfo*?
- (ii) Angi hvilke normalformer det er brudd på i *Ordreinfo*, og beskriv for hvert brudd hvilke FDer som bryter normalformen og hvorfor.
- (iii) Lovlige instanser av *Varegruppeinfo* i skjema (2) skal reflektere nøyaktig de lovlige instansene til relasjonene *Varegruppetilhørighet* og *Ansvarsområde* i skjema (1). Hvilke FDer og MVDer må i såfall gjelde i *Varegruppeinfo*?
- (iv) Angi hvilke normalformer MVDene fra (iii) bryter og hvorfor.

2 SQL

Ta utgangspunkt i skjemaet (1) og besvar følgende spørsmål med SQL.

- (i) Finn navn og adresse på alle kunder som har bestilt varer i varegruppen STRØMAGGREGATER.

- (ii) Lag en liste over navn, adresse og totalt beløp hver kunde har bestilt for i mai 2007. Kunder som ikke har bestilt noen varer i denne måneden, skal ikke være med på listen.
- (iii) Lag en liste over alle varene i varegruppen STRØMAGGREGATER. Hver linje i listen skal inneholde varenavn, enhetspris, det totale antall enheter bestilt av varen i år 2007 og antall forskjellige kunder som bestilte varen i 2007. Listen skal være sortert etter totalt antall bestilt slik at varen med flest bestilte enheter kommer først, og varer som ikke er bestilt av noen, kommer sist.

3 OQL

I vedlegg 1 finner du en beskrivelse av en objektorientert versjon av verktøygrossistdatabasen. Ta utgangspunkt i denne beskrivelsen og besvar spørsmålene (i)-(iii) fra oppgave 2 ved hjelp av OQL.

4 Implementasjon

I denne oppgaven skal du vise din forståelse av hvordan et databasesystem implementeres. Vi tar utgangspunkt i relasjonsdatabasen beskrevet i skjemaet (1).

Gitt følgende SQL-spørring som finner navn og adresser til enpersonsforetak som har bestilt varer i varegruppen GAFFELTRUCKER i 2007:

```

select  Kunde.navn, Kunde.adresse
from    Kunde, Ordre, Ordrelinje, Varegruppetilhørighet
where   Kunde.kundeID = Ordre.kundeID                and
          Ordre.ordreID = Ordrelinje.ordreID          and
          Ordrelinje.vareID = Varegruppetilhørighet.vareID and
          Kunde.ftype = 'E'                            and
          Varegruppetilhørighet.varegr = 'GAFFELTRUCKER' and
          Ordre.dato like '2007%'

```

Databasen har clustrede indekser på primærnøkklene. I tillegg er det indekser på attributtet *dato* i *Ordre* og på attributtet *foretaksnr* i *Kunde*.

4.1 Parsering

- (i) Bruk den enkle grammatikken i vedlegg 2 til å lage et parseringstre for spørringen ovenfor.
- (ii) Hvilke hovedoppgaver har preprosessoren?

4.2 Logisk spørreplan

Konverter parseringstreet i oppgave 4.1(i) til en logisk spørreplan i relasjonsalgebra (tegn uttrykkstreet). NB! Denne oppgaven skal løses uten optimering.

4.3 Optimering

Optimer den logiske spørreplanen i oppgave 4.2 (tegn det nye uttrykkstreet).

4.4 Eksekvering

Hvis datamengden er så stor at vi ikke kan holde hele datasettet i minnet samtidig, baserer vi eksekveringen av (fullrelasjons) operatorene på to grunnprinsipper.

- (i) Hvilke er disse, og hvordan fungerer de i korte trekk?
- (ii) I hvilke tilfeller vil du bruke de forskjellige metodene? Begrunn svaret.

4.5 Datalagring

For denne oppgaven skal vi konsentrere oss om følgende delmengde av skemaet i (1):

Kunde (*kundeID*, *navn*, *ftype*, *adresse*, *foretaksnr*)

Vare (*vareID*, *varenavn*, *enhpris*)

Ordrelinje (*ordreID*, *vareID*, *antbest*)

Ordre (*ordreID*, *kundeID*, *dato*)

Anta at vi har en disk med følgende spesifikasjoner for lagring av våre data:

| | |
|---------------------------|----------------------------|
| Diskplater: | 10 (med 2 overflater hver) |
| Spor: | 10.000 pr. overflate |
| Antall sektorer pr. spor: | 1000 (en ikke-sonet disk) |
| Byte pr. sektor: | 512 |
| Byte pr. "gap": | 64 |
| Gjennomsnittlig søketid: | 5 ms |
| Spor-spor søk: | 0,5 ms |
| Rotasjons hastighet: | 15.000 RPM |

Anta også at hver relasjon er lagret clustret på disken. Databasen inneholder følgende antall tupler:

| | |
|-----------------------------------|------------|
| Antall <i>Kunde</i> -tupler: | 100.000 |
| Antall <i>Vare</i> -tupler: | 10.000 |
| Antall <i>Ordrelinje</i> -tupler: | 10.000.000 |
| Antall <i>Ordre</i> -tupler: | 1.000.000 |

I tillegg gjelder følgende informasjon om diverse størrelser:

- Hver blokk har en "header" (hode) på 20 byte.
- Hver "record" (post) har et hode på 10 byte.
- Hvert attributt har følgende størrelse i antall bytes:

| <i>Kunde</i> | | <i>Vare</i> | | <i>Ordrelinje</i> | | <i>Ordre</i> | |
|---------------------|----|-------------------|----|-------------------|----|------------------|----|
| <i>kundeID</i> : | 16 | <i>vareID</i> : | 16 | <i>ordreID</i> : | 16 | <i>ordreID</i> : | 16 |
| <i>navn</i> : | 40 | <i>varenavn</i> : | 40 | <i>vareID</i> : | 16 | <i>kundeID</i> : | 16 |
| <i>ftype</i> : | 1 | <i>enhpris</i> : | 10 | <i>antbest</i> : | 10 | <i>dato</i> : | 10 |
| <i>adresse</i> : | 40 | | | | | | |
| <i>foretaksnr</i> : | 11 | | | | | | |

- Hva er diskens utnyttbare kapasitet? Ikke glem at hver plate har to overflater!
- Hvilke faktorer inngår i å aksessere en blokk på disken, og hva er gjennomsnittlig aksesstid for en vilkårlig 4 Kbyte blokk?
- Hvor stor plass trenger disse relasjonene på disken i tilfellet "unspanned" lagring (dvs. hvis ingen enkelt post er delt over flere blokker)?
- Hvor lang tid tar det å lese hele relasjonen *Ordrelinje* uavbrutt hvis vi antar vilkårlig plassering av data i diskblokker på disken?

- (v) Hvordan kan man optimere en slik lesing av en relasjon ved å forandre blokkplassering, og hva blir den nye tiden for å lese hele *Ordrelinje*?
NB: Skriv ned eventuelle antakelser du gjør!

5 Transaksjoner

Kundene bestiller varer over internett gjennom en “handlevognsapplikasjon”. Typisk vil kundene først gå gjennom en fase hvor de undersøker hvilke varer som fins, og deretter en fase hvor de velger ut hvilke varer, og hvor mange enheter av hver vare, som skal bestilles. Deretter ber de handlevognsapplikasjonen om å gjennomføre og bekrefte bestillingen.

Vi skal konsentrere oss om det som skjer med relasjonen *Lagerbeholdning*. For å gjennomføre en bestilling på vegne av en kunde, utfører applikasjonen en transaksjon som leser tuplene til de varene som skal bestilles, sjekker at det er nok enheter på lager av hver vare, og for de varene der det er nok enheter, bestemmer hvilke lagre varene skal plukkes fra og teller ned i *antall*. Dersom det for en vare ikke er nok enheter, foretas ingen bestilling av denne varen.

Hvis vi lar $r_i(A)$ og $w_i(A)$ betegne at en transaksjon T_i henholdsvis leser og skriver et tuppel A i *Lagerbeholdning*, så kan derfor to mulige transaksjoner i en gitt databasetilstand se slik ut:

$$\begin{aligned} T_1 &: r_1(A); r_1(B); w_1(A); w_1(B) \\ T_2 &: r_2(A); r_2(B); w_2(B) \end{aligned}$$

(T_1 bestiller enheter av to varer, representert ved henholdsvis A og B ; T_2 bestiller bare B fordi det viser seg å ikke være nok enheter av A .)

5.1 Serialiserbarhet

Betrakt følgende eksekveringsplan S_1 av T_1 og T_2 :

$$S_1 : r_1(A); r_2(A); r_1(B); w_1(A); w_1(B); r_2(B); w_2(B)$$

- (i) Vis at S_1 ikke er konfliktserialiserbar.
- (ii) Finn en antakelse om A og B som gjør S_1 serialiserbar. Er det rimelig å anta at S_1 er serialiserbar i handlevognapplikasjonen? Begrunn.

Vi skal nå se på en svakt endret applikasjon hvor grossisten er interessert i å reklamere på sin hjemmeside hvor godt besøkt siden er. Derfor vil hver transaksjon avslutte med å skrive et tidspunkt til hjemmesiden, dette tidspunktet illustrerer når hjemmesiden sist ble besøkt. Skrivning av tidspunkt gjøres ved operasjonen $w_i(Z)$. Betrakt følgende eksekveringsplan S_2 av ytterligere tre transaksjoner T_3 , T_4 og T_5 :

$$S_2 : \quad r_3(A); w_3(A); r_4(A); r_4(B); r_5(C); w_4(B); \\ w_4(Z); w_3(Z); w_5(C); w_5(Z)$$

- (iii) Tegn presedensgrafene til S_2 .
- (iv) Er S_2 konfliktserialiserbar? Begrunn.
- (v) Tegn polygrafene til S_2 .
- (vi) Er S_2 viewserialiserbar? Begrunn.

5.2 Samtidighetskontroll

5.2.1 Pessimistisk protokoll

Anta at vi har eksklusive låser på hvert tuppel i *Lagerbeholdning*. Låsene skal brukes på vanlig måte, ved at hver lese- og skriveaksjon skal ha en forutgående låseaksjon og en etterfølgende opplåsningsaksjon. Dessuten skal hver transaksjon benytte tofaselåsing (2PL).

La aksjonen $l_i(Y)$ bety at T_i tar låsen på Y og $u_i(Y)$ at T_i frigir låsen på Y .

- (i) Legg inn aksjoner av formen $l_i(Y)$ og $u_i(Y)$ i hver av T_1 og T_2 slik at de oppfyller reglene for bruk av låsene under 2PL, og samtidig frigir låser så snart som mulig.
- (ii) Beskriv hva som skjer hvis vi prøver å utføre aksjonene i de resulterende transaksjonene slik at lese/skriveaksjonene utføres mest mulig i samsvar med rekkefølgen angitt av S_1 .

Anta så at vi på hvert tuppel i *Lagerbeholdning* har to låser – en delt (S-lås, shared lock) og en eksklusiv (X-lås), der en S-lås kan oppgraderes til (byttes ut med) en X-lås ved behov. Låsene skal forøvrig brukes som vanlig for S/X-låser og i henhold til 2PL.

La aksjonene $ls_i(Y)$ og $lx_i(Y)$ bety at T_i tar henholdsvis S-låsen og X-låsen på Y , og $u_i(Y)$ at T_i frigir alle sine låser på Y .

- (iii) Legg inn aksjoner av formen $ls_i(Y)$, $lx_i(Y)$ og $u_i(Y)$ i hver av T_1 og T_2 slik at de oppfyller reglene for bruk av låsene under 2PL, og slik at transaksjonene ikke benytter X-låser mer enn strengt nødvendig (dvs. de benytter oppgradering der dette er mulig). Låser skal frigis så snart som mulig.
- (iv) Beskriv hva som skjer hvis vi prøver å utføre aksjonene i de resulterende transaksjonene slik at lese/skriveaksjonene utføres mest mulig i samsvar med rekkefølgen angitt av S_1 .

5.2.2 Optimistisk protokoll

Vi skal så se på hva som skjer hvis vi bruker en tidsstemplingsprotokoll. Anta at T_1 får tidsstempelet t_1 og T_2 tidsstempelet t_2 , hvor $t_1 < t_2$.

- (i) Beskriv hva som skjer med T_1 og T_2 hvis vi prøver å utføre aksjonene i rekkefølgen angitt av S_1 .

Anta videre at T_3 , T_4 og T_5 får tidsstemplene t_3 , t_4 og t_5 hvor $t_3 < t_4 < t_5$. Anta videre at T_4 aborterer (må ruller tilbake) etter at alle aksjonene dens er utført.

- (ii) Beskriv hva som skjer med T_3 og T_5 hvis vi prøver å utføre aksjonene i rekkefølgen angitt av S_2 .

6 Logging

- (i) Beskriv kort prinsippet bak undo-logging. Hva slags log-records trenger man? Når skal de forskjellige typene log-recorder skrives til disk?
- (ii) Beskriv kort forskjellen mellom undo- og redo-logging.

7 RAID 6

En vanlig type diskkabinetter (i 2008) inneholder 14 fysiske disker nummerert fra 1 til 14. Anta at vi organiserer disse som RAID 6 med disk 1, 2, 4 og 8 som Hammingkodede paritetsdisker. De resterende 10 diskene er altså vanlige datadisker.

Forklar hvordan systemet kan rekonstruere disk 3 og 11 hvis disse kræsjer samtidig og må skiftes ut.

Vedlegg 1 – Objektorientert versjon av verktøy-grossistdatabasen

```
class Ansatt (extent ansatte key ansattID)
{
  attribute string ansattID;
  attribute string navn;
  attribute string adresse;
  relationship Set<Varegruppe> ansvarsområder
    inverse Varegruppe::ansvarlige;
}

class Kunde (extent kunder key kundeID)
{
  attribute string kundeID;
  attribute string navn;
  attribute string adresse;
  relationship Foretaksnummer foretaksnr
    inverse Foretaksnummer::foretaksinfo;
  relationship Set<Ordre> ordrer
    inverse Ordre::kunde;
}

class Foretaksnummer (extent foretaksnumre key foretaksnr)
{
  attribute integer foretaksnr;
  attribute character ftype;
  relationship Kunde foretaksinfo
    inverse Kunde::foretaksnr;
}
```

```

class Vare (extent varer key vareID)
{
    attribute string vareID;
    attribute string varenavn;
    attribute integer enhpris;
    relationship Set<Ordrelinje> ordrer
        inverse ordrelinje::vare;
    relationship Set<Lagerbeholdning> beholdning
        inverse Lagerbeholdning::vare;
    relationship Set<Varegruppetilhørighet> varegrupper
        inverse Varegruppetilhørighet::varer;
}

class Lagerbeholdning (extent lagerbeholdninger)
{
    attribute integer antall;
    relationship Lager lager
        inverse Lager::beholdning;
    relationship Vare vare
        inverse Vare::beholdning;
}

class Lager (extent lagere)
{
    attribute string lagernavn;
    relationship Set<Lagerbeholdning> beholdning
        inverse Lagerbeholdning::lager;
}

class Varegruppetilhørighet (extent varegruppetilhørigheter)
{
    attribute string varegr;
    relationship Set<Ansatt> ansvarlige
        inverse Ansatt ansvarsområder;
    relationship Set<Vare> varer
        inverse Vare::varegrupper;
}

```

```
class Ordre (extent ordrer key ordreID)
{
  attribute string ordreID;
  attribute date dato;
  relationship Kunde kunde
    inverse Kunde::ordrer;
  relationship Set<Ordrelinje> ordrelinjer
    inverse Ordrelinje::ordre;
}
```

```
class Ordrelinje (extent ordrelinjer)
{
  attribute integer antall;
  relationship Ordre ordre
    inverse Ordre::ordrelinjer;
  relationship Vare vare
    inverse Vare::ordrer;
}
```

Vedlegg 2 – Grammatikk for parsing av spørsmål

`<query> ::= <SFW>`

`<SFW> ::= SELECT <selList> FROM <fromList> WHERE <condition>`

`<selList> ::= <attribute>`

`<selList> ::= <attribute>, <selList>`

`<fromList> ::= <relation>`

`<fromList> ::= <relation>, <fromList>`

`<condition> ::= <condition> AND <condition>`

`<condition> ::= <attribute> = <attribute>`

`<condition> ::= <attribute> = <pattern>`

`<condition> ::= <attribute> LIKE <pattern>`

Elementære syntaktiske kategorier som `<attribute>`, `<relation>` og `<pattern>` har ingen regler, men oversettes med henholdsvis navnet på attributtet, navnet på relasjonen og en streng i anførselstegn.

Slutt på obligatorisk oppgave 2