INF3100 V2016
Obligatorisk oppgave nr. 2

Oppgavesettet skal i utgangspunktet løses av grupper på to og to studenter. Vi godkjenner også individuelle besvarelser, men oppfordrer dere altså til heller å finne en å samarbeide med. Vi godkjenner ikke grupper med mer enn to studenter.

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

`www.uio.no/studier/admin/obligatoriske-aktiviteter/mn-ifi-oblig.html`

Enhver innlevering av besvarelse på en obligatorisk oppgave tas som en bekreftelse på at retningslinjene er lest og forstått.

Innleveringsfrist: Fredag 6. mai kl. 23.59.

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

Oppgave 1–5 Eksamen 2014

Løs oppgave 1 (FDer), oppgave 2 (SQL), oppgave 3 (relasjonsalgebra), oppgave 4 (transaksjonsprotokoller) og oppgave 5 (RAID-teknologier) fra eksamen i INF3100 våren 2014. (Scenario: Matrikkelen i Norge.)

Oppgave 6 Spørsmålsoptimalisering

Folkeregisteret inneholder informasjon om alle innbyggere i Norge og hvor de bor. Alle personer har et entydig 11-sifret fødselsnummer `fnr`. Når noen flytter, må de sende inn en flyttemelding til folkeregisteret. Flyttemeldingen kan omfatte mer enn én person, f.eks. hvis en hel familie flytter. Hver flyttemelding er identifisert ved et løpenummer `mid`. Forøvrig inneholder den

informasjon om flyttedato, gammel og ny adresse. Anta at denne informasjonen er organisert i følgende relasjoner der primærnøklene er understreket:

```
Person(fnr, fornavn, etternavn)
Flyttemelding(mid, flyttedato, fraadr, tiladr)
FlyttetPerson(mid, fnr)
```

Betrakt følgende SQL-spørring som finner alle flyttemeldingene (i form av løpenummer og flyttedato) til Jo Å:

```
SELECT  Flyttemelding.mid, Flyttemelding.flyttedato
FROM    Person, Flyttemelding, FlyttetPerson
WHERE   Person.fnr = FlyttetPerson.fnr AND
        Flyttemelding.mid = FlyttetPerson.mid AND
        Person.fornavn = 'Jo' AND
        Person.etternavn = 'Å';
```

6a. Bruk den enkle grammatikken under til å lage et parseringstre for SQL-spørringen.

```
<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>
<condition> ::= <attribute> IS NULL
```

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

6b. Konverter parseringstreet i 6a til en logisk spørreplan i relasjonsalgebra (tegn uttrykkstreet).

6c. Optimer den logiske spørreplanen i 6b hvis den ikke alt er på optimal form (tegn det nye uttrykkstreet).

Oppgave 7 Transaksjonsprotokoller

I oppgavene 7–9 skal vi benytte transaksjonene

$$T_1 = r_1(a); r_1(b); w_1(b); r_1(c); w_1(c)$$
$$T_2 = r_2(b); r_2(d); w_2(d)$$
$$T_3 = r_3(d); r_3(a); w_3(a).$$

Anta at vi har en eksekveringsplan

$$S_1 = r_1(a); r_3(d); r_1(b); r_2(b); w_1(b); r_3(a); w_3(a); r_1(c); w_1(c); r_2(d); w_2(d)$$

7a. Avgjør om S_1 er konfliktserialiserbar.

Anta at systemet tilbyr lese- og skrivelåser. La $sl_i(y)$ og $xl_i(y)$ betegne at en transaksjon T_i tar henholdsvis en leselås (delt lås) og en skrivelås (eksklusiv lås) på elementet y. La $u_i(y)$ betegne at T_i frigir låsen på y.

Vi skal se på bruk av oppgraderingslåser:

- Hvis en transaksjon skal lese et element, ber den om en leselås på elementet. Hvis den senere skal skrive elementet, ber den om en skrivelås på elementet. (Når skrivelåsen tildeles, byttes leselåsen ut med skrivelåsen.)
- Hvis en transaksjon bare skal skrive et element (og ikke lese det først), ber den om en skrivelås på elementet.

7b. Sett inn oppgraderingslåser i transaksjonene T_1, T_2 og T_3 i henhold til strikt tofaselåsing (strict 2PL).

7c. Beskriv hva som skjer hvis lese- og skriveoperasjonene så langt som mulig utføres i rekkefølgen angitt i S_1.

Oppgave 8 Vranglås

Hvis transaksjonene T_1, T_2 og T_3 bruker lese- og skrivelåser uten oppgradering, vil f.eks. eksekveringsplanen

$$S_2 = r_1(a); r_2(b); r_3(d); r_1(b); w_1(b); r_2(d); w_2(d); r_3(a); w_3(a); r_1(c); w_1(c)$$

gi en vranglås fordi følgende skjer:

1. T_1 starter.
2. T_1 ber om og får en leselås på a.
3. T_2 starter.
4. T_2 ber om og får en leselås på b.
5. T_3 starter.
6. T_3 ber om og får en leselås på d.
7. T_1 ber om en skrivelås på b, men må vente fordi T_2 har leselås på b.
8. T_2 ber om en skrivelås på d, men må vente fordi T_3 har leselås på d.
9. T_3 ber om en skrivelås på a, men må vente fordi T_1 har leselås på a.

8a. Anta at systemet bruker deadlock-protokollen vent-dø. Beskriv hvilken transaksjon som blir abortert/rullet tilbake, og hvorfor.

8b. Anta at systemet bruker deadlock-protokollen skad-vent. Beskriv hvilken transaksjon som blir abortert/rullet tilbake, og hvorfor.

Oppgave 9 Logging

Det semantiske innholdet av T_1 består av følgende operasjoner (x, y og z er lokale arbeidsvariable for T_1 og skal derfor ikke logges):

$$T_1 : \quad x := a; y := b; y := x + y; b := y; z := c; z := z + 1; c := z;$$

Anta at vi initielt har verdiene $a = 13$, $b = 17$ og $c = 19$.

9a. Beskriv postene i undo-loggen for transaksjonen T_1.

9b. Når skal de forskjellige typene loggposter skrives til disk ved undo-logging?

Slutt på obligatorisk oppgave 2