INF3100 V2018
Obligatorisk oppgave nr. 2

Oppgavesettet skal løses og leveres individuelt.

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 4. mai kl. 23.59.

Fristen er absolutt. Alle spørsmålene må besvares for å få godkjent besvarelsen.

Oppgave 1 Transaksjonsprotokoller

I oppgavene 1 og 2 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

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

1a. 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.

1b. Sett inn oppgraderingslåser i transaksjonene T_1, T_2 og T_3 i henhold til strikt tofaselåsing (strict 2PL).

1c. Beskriv hva som skjer hvis lese- og skriveoperasjonene så langt som mulig utføres i rekkefølgen angitt i S_1.

Oppgave 2 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.

2a. Anta at systemet bruker deadlock-protokollen vent-dø. Beskriv hvilken transaksjon som blir abortert/rullet tilbake, og hvorfor.

2b. Anta at systemet bruker deadlock-protokollen skad-vent. Beskriv hvilken transaksjon som blir abortert/rullet tilbake, og hvorfor.

Oppgave 3 Isolasjonsnivåer i postgres

Denne oppgaven vil krever litt googling og manual-lesing. I denne oppgaven skal vi se på hva som skjer i postgres under isolasjonsnivåene repeatable read og serializable. Til dette skal vi bruke følgende tabell:

```
CREATE TABLE dots (
    id INT NOT NULL PRIMARY KEY,
    color TEXT NOT NULL
  );

INSERT INTO dots
  WITH x(id) AS (SELECT generate_series(1,10))
  SELECT id, CASE WHEN id % 2 = 1 THEN 'black'
    ELSE 'white' END FROM x;
```

Anta så at vi kjører følgende to transaksjoner parallellt:

```
--T1
begin;
update dots set color = 'black'
  where color = 'white';
commit;

--T2
begin;
update dots set color = 'white'
  where color = 'black';
commit;
```

Man kan teste dette ved å logge seg inn i postgres i to faner, og kjøre en transaksjon i hver (vent med commit). Man kan sette isolasjonsnivå for transaksjonene eller for hele sesjonen (to innlogginger betyr to sesjoner, sett riktig nivå i begge).

3a. Hva vil skje hvis disse transaksjonene kjører parallellt under isolasjonsnivå SERIALIZABLE, og hvorfor?

3b. Hva vil skje hvis disse transaksjonene kjører parallellt under isolasjonsnivå REPEATABLE READ, og hvorfor?

Oppgave 4 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 $c = 19$.

4a. Beskriv postene i undo-loggen for transaksjonen T_1.

4b. Når skal de forskjellige typene loggposter skrives til disk ved undo-logging?

4c. Beskriv postene i undo/redo-loggen for transaksjonen T_1.

4d. Når skal de forskjellige typene loggposter skrives til disk ved undo/redo-logging?

Oppgave 5 Query optimization

La $R(x, y)$ være en tabell med 100000 tupler. Anta at vi har blokker a 500 tupler, og at tabellen ligger tettpakket (i så få blokker som mulig) på disk. Anta også at vi har en indeks på x, som ligger i minnet (så vi trenger ikke å ta med kostnaden av å lese indeksfilen).

Anta at det ikke er forskjell på sekvensiell og random lesing fra disk (med andre ord, å lese en blokk koster 1 IO). Vi skal først se på IO-kostnaden til seleksjonen $\sigma_{x=a}R$ (det å finne og lese inn alle tupler som tilfredsstiller denne) med og uten bruk av indeksen.

Vi kan gjøre følgende observasjoner: Dersom $V_x(R)$ (antall distinkte verdier for x i R) er 1, så kan vi slå opp i indeksen alene for å sjekke om a finnes i det hele tatt. Hvis nei, returnerer vi ingen tupler. Hvis ja, så skal vi returnere hele tabellen, og da er det ingen vits i å bruke indeksen til å lese inn blokkene. Hvis $V_x(R)$ derimot er lik 100000, så er x en nøkkel, og vi ute etter maks en blokk. Denne kan vi billig finne via indeksen. Vi konkluderer med at den estimerte kostnaden til å bruke indeksen her blir lavere jo høyere $V_x(R)$ er.

Finn det høyeste antall distinkte verdier for x i R ($V_x(R)$) slik at det *ikke* lønner seg å bruke indeksen (med hensyn på estimert kostnad i worst-case) gitt at indeksen

5a er en tett indeks og R ikke ligger sortert på x?

Slutt på obligatorisk oppgave 2