NTNU
Det skapende universitet

Institutt for datateknikk og informasjonsvitenskap

Løsningsskisse til Eksamensoppgave i TDT4145 Datamodellering og databasesystemer

Eksamensdato: 26. mai 2014

Eksamenstid (fra-til): 09:00 - 13:00

Hjelpemiddelkode/Tillatte hjelpemidler:

D – Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Versjon 5. juni 2014.

Oppgave 1 – Datamodeller (20 %)

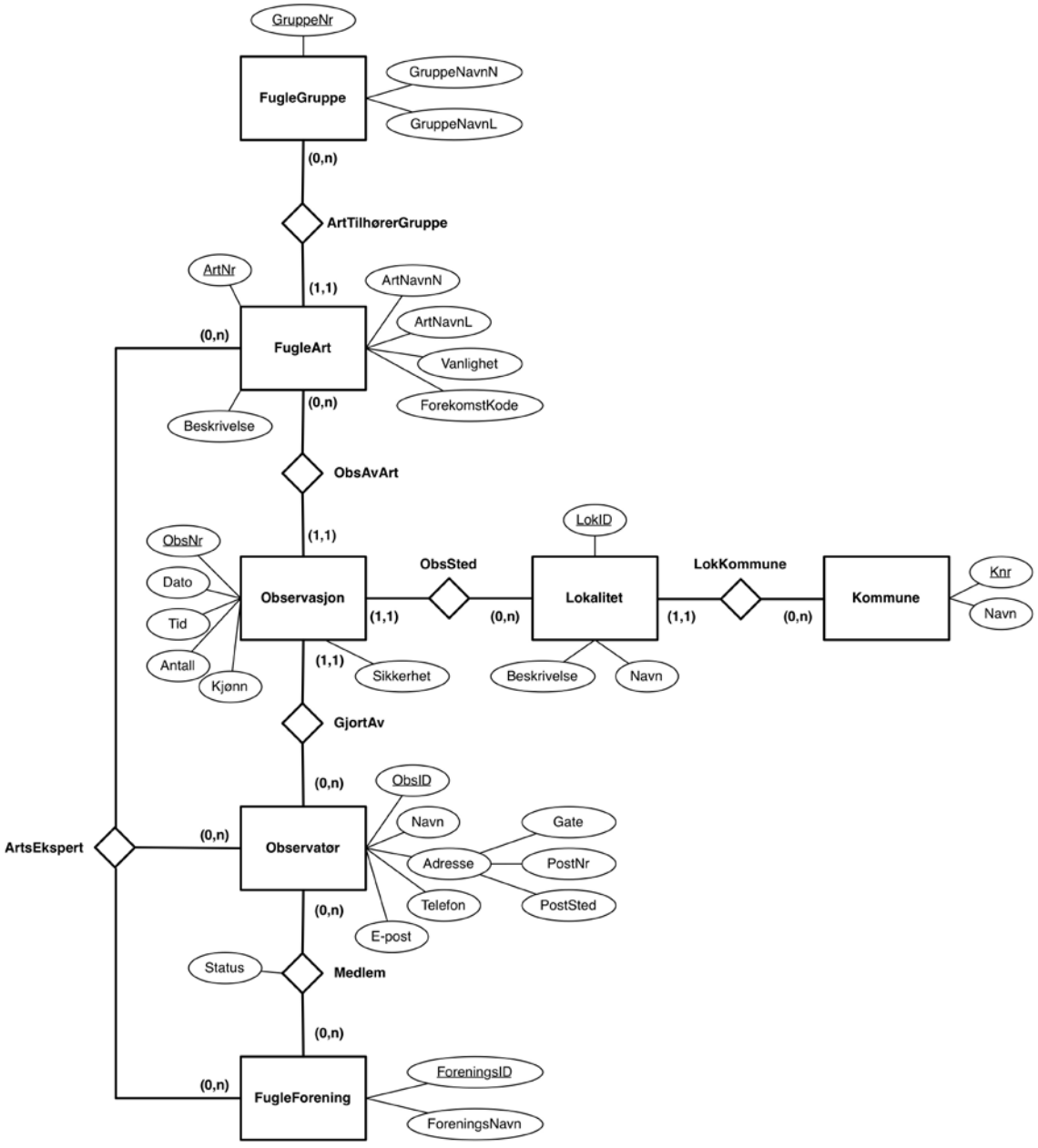
a) Forekomsten til en tertiær relasjonsklasse kan ses på som en mengde av tre-tupler av typen (a_i, b_j, c_k), der a_i er en entitet i A, b_j er en entitet i B og c_k er en entitet i C. De spesifiserte restriksjonene har følgende konsekvenser i forhold til å begrense mulige forekomster av 3-tupler i R:

- **(0,n) ved A**: En entitet i A trenger ikke å være med i noe 3-tuppel, men kan være med i vilkårlige mange. *Relasjonen er ikke obligatorisk for entiteter i A og entitetene kan har relasjonen til et vilkårlig antall forskjellige kombinasjoner av B- og C-entiteter*.
- **(1,m) ved B**. Alle entiteter i B må være med i minst ett 3-tuppel og kan være med i vilkårlig mange. *Relasjonen er obligatorisk for entiteter i B og entitetene kan ha relasjonen til et vilkårlig antall forskjellige kombinasjoner av A- og C-entiteter*.
- **(1,2) ved C**. En entitet i A må være med i minst ett 3-tuppel og kan ikke være med i flere enn to 3-tupler. *Relasjonen er obligatorisk for entiteter i C og hver entitet kan ha relasjonen til høyst to forskjellige kombinasjoner av A- og B-entiteter*.

b) ER-diagrammet på neste side viser en mulig ER-modell. Det vil være like riktig å spesifisere attributtene inne i boksene, slik det er gjort i løsningen til oppgave 2a. For flere av attributtklassene vil det i "virkeligheten" være mange aktuelle attributter, men ut over det som fremgår av oppgaveteksten, er det ikke behov for å ta med flere enn strengt nødvendig.

I løsningen har vi gjort følgende forutsetninger:

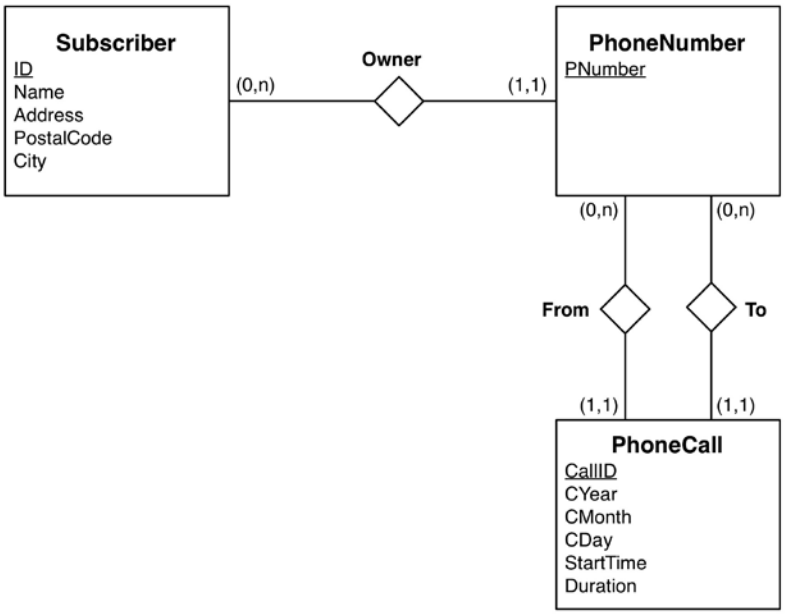
- FugleGrupper kan eksistere uten noen tilhørende FugleArt.
- For FugleArt har vi innført ArtNr som nøkkel. Identifikatorbehovet kan løses på andre måter.
- Vi har innført et unikt ObsNr for å ha en nøkkel i Observasjon.
- Vi har innført en ObsID for Observatør for å ha en nøkkel for denne klassen. Observatører kan registreres uten at de har gjort noen Observasjon.
- For FugleForening har vi tatt med ForeningsID som nøkkel og ForeningsNavn.
- For Lokalitet (observassjonssted) har vi innført nøkkelen LokId og attributtene navn og beskrivelse. Siden Lokalitet ligger i en bestemt kommune, kan den evt. gjøres til en svak klasse under Kommune, med de følger dette vil ha for å kunne ha en identifikator for klassen.
- For Kommune har vi tatt med et unikt Knr og navn på kommunen.



Oppgave 2 – Relasjonsalgebra og SQL (20 %)

I SQL-oppgavene er svarene gitt med join spesifisert i FROM-delen. Det skal ikke trekkes for løsninger som spesifiserer join-betingelser i WHERE-delen.

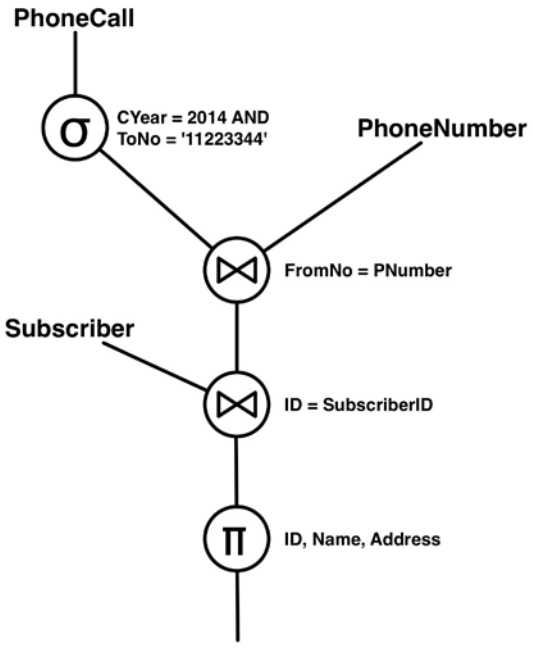
a) ER-diagrammet under viser en ER-modell for relasjonsskjemaet.



Forutsetninger:

- En Subscriber trenger ikke å ha registrert et PhoneNumber.

b) Relasjonsalgebra:



c)
```
SELECT PNumber
FROM PhoneNumber INNER JOIN Subscriber ON SubscriberID = ID
WHERE Address LIKE 'Buckhaugen%';
```

d)
```
SELECT City, COUNT(*)
FROM Subscriber
GROUP BY City
ORDER BY COUNT(*) DESC;
```

e)
```
UPDATE Subscriber SET
       Address = 'Buckhaugen 99',
       PostalCode = 7046,
       City = 'Trondheim'
WHERE ID = 100;
```

f)
```
SELECT PNumber, ID, Name, Address
FROM Subscriber INNER JOIN PhoneNumber ON ID = SubscriberID
WHERE PNumber NOT IN
      (SELECT FromNo FROM PhoneCall WHERE CYear = 2014)
ORDER BY ID ASC, PNumber ASC;
```

Oppgave 3 – Teori (20 %)

a) En mengde attributter, K, er en *kandidatnøkkel* for en tabell R dersom:

- K er en supernøkkel ($K^+ = R$) og
- K er minimal (ingen ekte delmengde av K er en supernøkkel).

b) Siden A ikke er på noen høyreside i F må den være med i alle kandidatnøkler. Vi prøver alle muligheter som oppstår ved å kombinere A med ett annet attributt:

- $A^+ = AB^+ = AB \subset R$
- AC+ = ABCD = R
- AD+ = ABCD = R

Det er ingen andre muligheter som vil være minimale. Kandidatnøklene er AC og AD.

c) Siden kandidatnøklene er AC og AD er nøkkelattributtene (prime) ACD. B er det eneste ikke-nøkkel-attributtet (non-prime). På grunn av A -> B, vil B være delvis avhengig av begge kandidatnøklene. Andre normalform forbyr ikke-nøkkel-attributter å være delvis avhengig av en eller flere kandidatnøkler. Denne tabellen er derfor *ikke* på 2NF. Første normalform er derfor den høyeste normalformen som er oppfylt.

d) En mulig dekomponering er: R_1(A, B), R_2(A, C) og R_3(C, D). A er primærnøkkel i R_1. AC er primærnøkkel i R_2. C og D er kandidatnøkler i R_3. Dekomponeringer vurderes etter fire forhold:

- **Attributtbevaring:** $R = R_1 \cup R_2 \cup R_3$ så det er ivaretatt.
- **FD-bevaring:** A -> B ivaretas i R_1, C - > D og D -> C ivaretas i R_3. Det betyr at alle FD-er i F er ivaretatt og vi har FD-bevaring.

- **Tapsløst-join:** R_1 og R_2 joiner tapsløst fordi det felles attributtet A er (super-)nøkkel i R_1. Resultatet joiner tapsløst med R_3 til utgangspunktet (R) siden det felles attributtet (C) er en (super-)nøkkel i R_3. Vi har med andre ord oppfylt kravene til tapsløst-join-egenskapen. Egenskapen kan alternativt vises ved å bruke matrisemetoden (algoritme 15.3 i læreboken).
- **Normalform:** R_1 har F_1 = {A -> B} og er på BCNF siden A er en (super-)nøkkel i R_1. F_2 = ∅ slik at R_2 er på BCNF. R_3 har G_3 = { C -> D, D -> C } og er på BCNF siden begge venstresidene er (super-)nøkler i tabellen.

Dekomponeringen har god kvalitet siden alle kriterier er oppfylt.

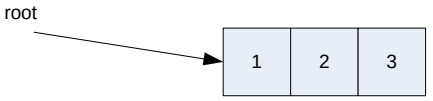
e) Når A ->> B gjelder betyr det at mengden B-verdier som er knyttet til en A-verdi vil være den samme uansett C-verdi. I den oppgitte tabellforekomsten er det bare en rad med a_2 så den gir ingen problemer. Det er 3 rader med a_1. a_1 er kombinert med b_1 i raden med c_1 og med b_1 og b_2 i radene med c_2. Slik det står er b-verdiene avhengig av c-verdiene. Vi trenger derfor raden (a_1, b_2, c_1) for at A –>> B skal være oppfylt. X = a_1, Y = b_2 og Z = c_1.

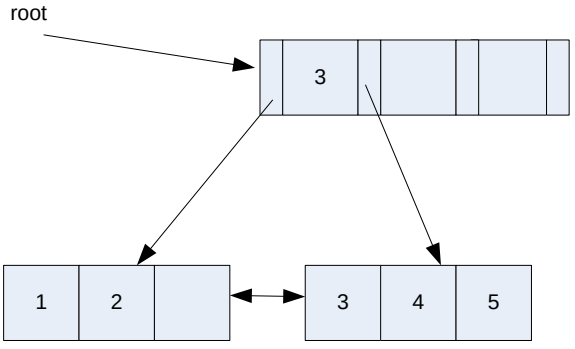
f) Ikke-trivielle MVD-er opptrer alltid i par, slik at når A ->> C gjelder i R så vil også A ->> B gjelde. Det betyr at alle andre forslag enn løsningen på forrige oppgave, vil sørge for at A ->> C *ikke* kan gjelde for tabellen.

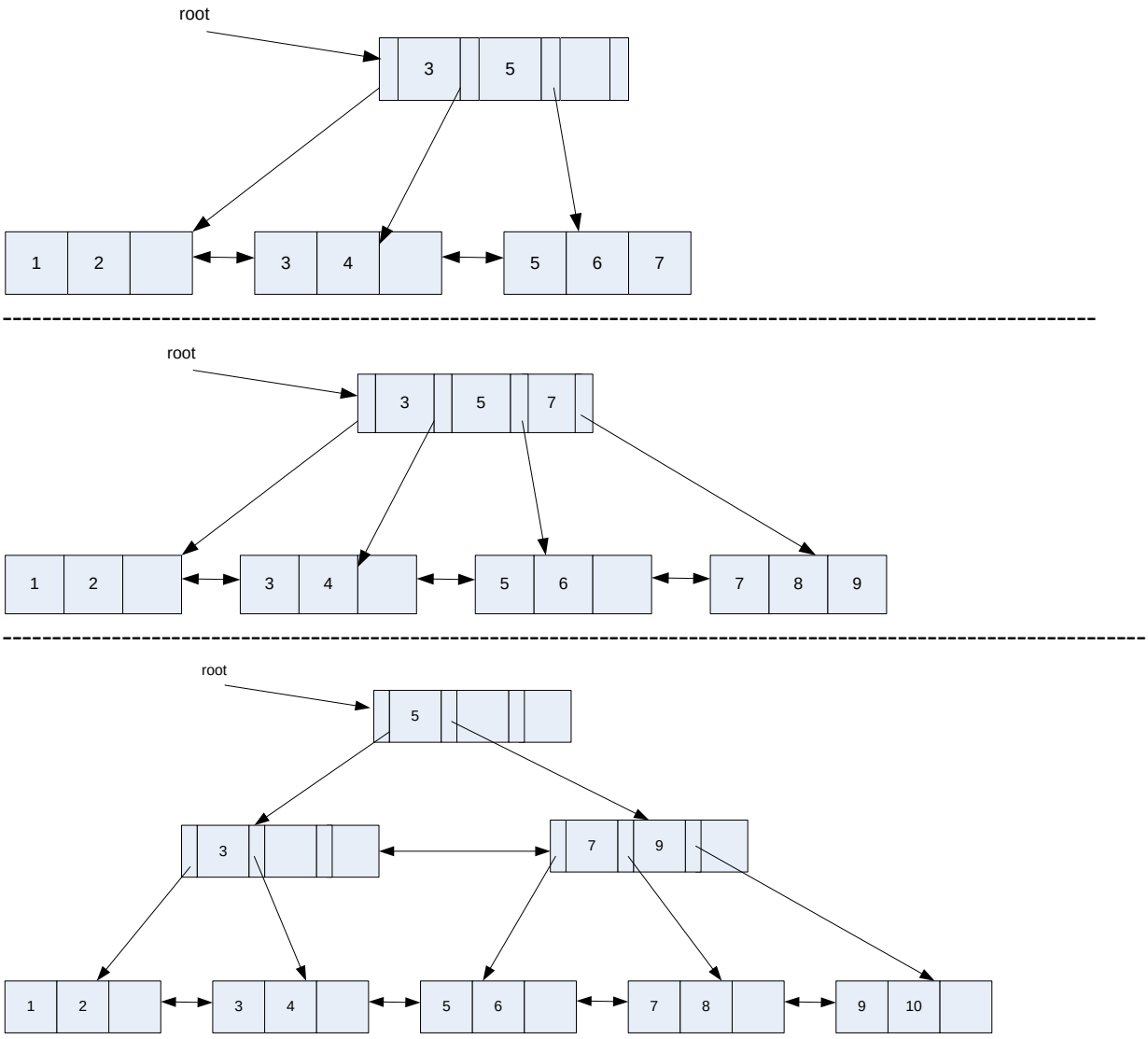
Oppgave 4 – B+-trær (10 %)

Vi skal sette inn følgende nøkler i et B+-tre i den gitte rekkefølgen: 1, 2, 3, 4, 5, 6, 7, 8, 9 og 10. Det er plass til maksimalt tre nøkler i hver blokk. Vis tilstanden til B+-treet hver gang du skal til å splitte en blokk. Vis også hvordan B+-treet ser ut til slutt.

Løsningsforslag: Vi benytter oss av løsningen som er beskrevet i notatet om B+-trær. Her splittes blokker ved at minste nøkkel i den nye høyreblokka settes inn på nivået over. I Elmasri & Navathe settes største nøkkel i venstreblokka inn på nivået over. Begge løsningene godtas, men samme splittemetode må brukes ved alle blokksplittene.







Oppgave 5 – Lagring og queryutføring (15 %)

a) (10 %) Gjør et estimat på hvor mange blokker som aksesseres (leses) ved de følgende SQL-setningene:

i) INSERT INTO Student VALUES (12123,'Hansen','Hans','hans@email.org',2013);
Løsningforslag: Her settes raden inn som en post i heapfila og det settes inn en post i indeksen også. For Heapfila sier vi at man lese siste blokk, dvs. 1 lesing. For B+-treet leses en blokk per nivå, dvs. 3 blokker til sammen. **Totalt antall lesinger er da 4**.
Hvis du tar med skrivinger, blir det en skriving for heapfila og en for B+-treet, dvs. totalt antall I/O er da 4 lesinger og 2 skrivinger, dvs. 6.

ii) SELECT lastname, firstname, email, startyear FROM Student WHERE lastname='Hansen';
Løsningsforslag: Her må det antas hvor mange 'Hansen' det finnes. Hvis det kun

er en Hansen, blir det 3 lesinger (nedover) i B+-treet og 1 lesing i Heapfila, dvs. **totalt 4 lesinger**.
Hvis vi antar to stk. Hansen, blir det 3 + 2, dvs**. totalt 5 lesinger**. Hvis vi antar at 1% av studentene heter Hansen, må vi lese 5 blokker sidevegs i B+treet i tillegg (1% av 500 er 5), og så slå opp 200 ganger i heapfila, dvs. 7 for B+treet og 200 for Heapfila, totalt 207 lesinger. Å scanne heapfila direkte uten å bruke B+-treet er 1000 lesinger.

iii) SELECT * FROM Student;
Løsningsforslag: Her er det lurt å ikke bruke indeksen, men å scanne Heapfila direkte, dvs. **1000 lesinger**. Hvis du gjør bruk av B+-treet her, vil du potensielt få 20000 lesinger pluss lesing av B+-treet, dvs. 20502 lesinger.

iv) SELECT DISTINCT lastname FROM Student ORDER BY lastname;
Løsningsforslag: Her er det lurt å bruke det ferdig sorterte B+-treet, dvs. du leser nedover treet og bortover: dvs. totalt 2 + 500 lesinger = **502 lesinger**. Hvis du leser heapfila, må du sortere fila. Kanskje noen studenter svarer med flettesortering av heapfila?

b) (5 %) Hvis den samme tabellen hadde fått kravet om at 'studno' skal være PRIMARY KEY, hvordan ville du ha lagret tabellen? Begrunn svaret.
Løsningsforslag: Her må det lages en **unik indeks på studno**, f.eks. ved å bruke et **Clustered B+-tre med studno som søkenøkkel**. Det er også mulig å bruke en unclustered indeks på studno.

Oppgave 6 – Transaksjoner - vranglåser (5 %)

Når vi bruker låser av dataelementer kan transaksjoner oppleve vranglåser. Hvordan løses vranglåser mellom transaksjoner?
Løsningsforslag:
1. **Oppdagelse**: Konstruer **wait-for-grafer** og søk etter sykler. Aborter en transaksjon og se om sykelen blir borte.
2. **Timeout**: La hver transaksjon ha en timeout-verdi. Start tilbakerulling av transaksjonen når timeout'en går.
3. **Unngåelse** er ikke i seg selv pensum i år. Men **konservativ 2PL** er pensum, hvor alle låser settes på forhånd, og da vil vi unngå vranglåser.

Oppgave 7 – Transaksjoner - recovery (10 %)

a) Anta T1, T2 og T3 er alle transaksjonene som finnes. Hvordan ser transaksjonstabellen ut etter analysefasen av recovery?
Løsningsforslag: Analysen starter ved siste sjekkpunktloggpost hvor transaksjonstabellen leses inn og så scannes loggen forover og transaksjonstabellen oppdateres. T3 er committet før sjekkpunktloggposten lages. Da vil transaksjonstabellen etter recovery bli slik:

TransactionId	**LastLSN**	**Status**
T2	112	Commit
T1	110	In progress
T3	105	Commit

Det kan diskuteres om T3 hører hjemme her, i og med at T3 committet før sjekkpunktet ble lagd. Men figur 22.5 i læreboka (Elmasri & Navthe) tar med committede transaksjoner i sjekkpunktet. I den gamle læreboka (Ramakrishnan & Gehrke) var det en ekstra end_transaction-loggpost som fjernet transaksjonen.

b) Hva er verdiene til A, B, C og D etter at recovery (alle faser) er ferdig?
Løsningsforslag: Det er kun T1 som rulles tilbake, derfor vil verdiene være **A = 33, B = 18, C = 40, D = 26**.