

Løsningsskisse til Eksamensoppgave i TDT4145 Datamodellering og databasesystemer

Eksamensdato: 1. juni 2015

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

Hjelpemiddelkode/Tillatte hjelpemidler:

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

Versjon 22. april 2016.

Oppgave 1 – Datamodeller (20 %)

a) **Foretrukket løsning:**

Vehicle(RegNo, Producer, Model, Weight, Length, Height)

PassengerCar(RegNo, Seats)

Bus(RegNo, PassengerSeats, DriversLicenceRequirement, CargoVolume)

Truck(RegNo, CargoVolume, MaxCargoWeight, Class)

Alternativ løsning:

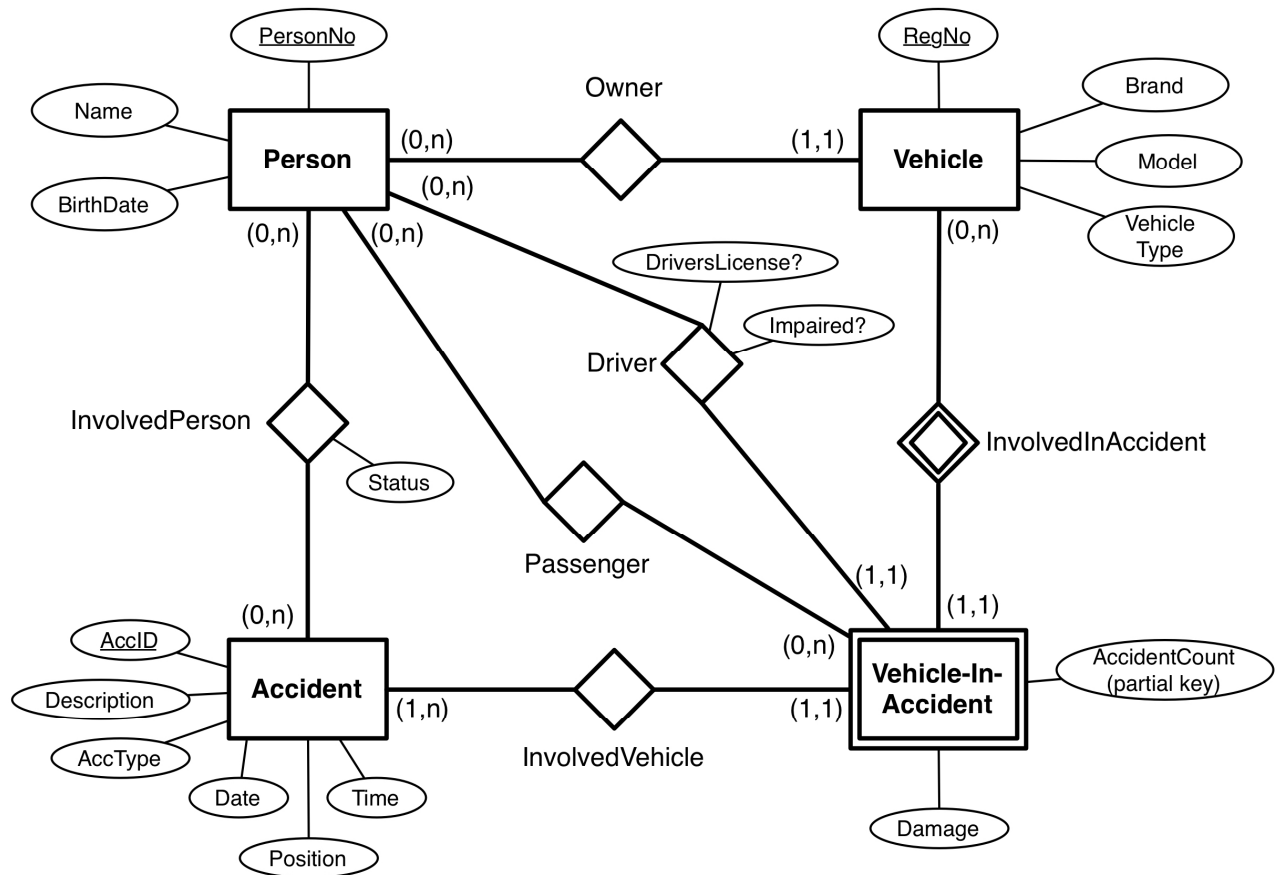
Vehicle(RegNo, , Producer, Model, Weight, Length, Height, Seats, PassengerSeats, DriversLicenceRequirement, CargoVolume, MaxCargoWeight, Class, TypeOfVehicle)

I denne løsningen vil attributtet TypeOfVehicle kunne ha verdiene: 'Car', 'Bus', 'Truck' eller NULL.

Vi anser at den alternative løsningen er dårligere fordi det er vanskelig å holde oversikt over hvilke attributter som hører til hvilken underklasse av kjøretøy og fordi det nødvendigvis vil bli mange NULL-verdier i tabellen.

Siden spesialiseringen er delvis kan vi *ikke* bruke en løsning uten en egen tabell for superklassen Vehicle, med overføring av de felles attributtene til underklasse-tabellene.

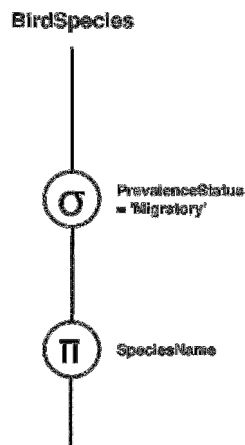
b) ER-diagrammet under viser en mulig datamodell. Det vil være like riktig å spesifisere attributtene inne i boksene.



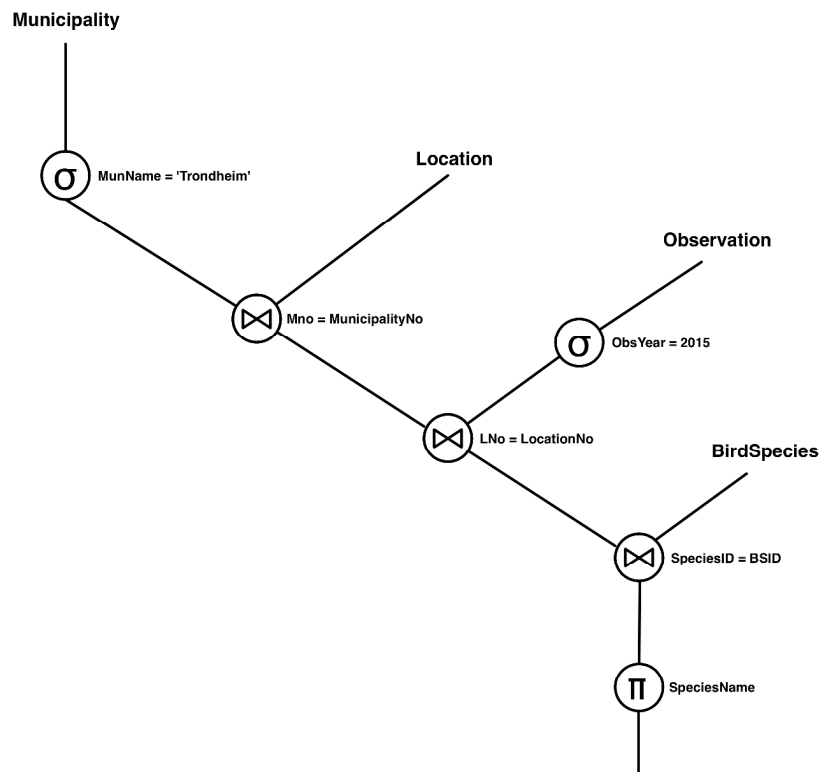
Oppgave 2 – Relasjonsalgebra og SQL (20 %)

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) Relasjonsalgebra:



b) Relasjonsalgebra:



- c) `SELECT SpeciesName
FROM BirdGroups INNER JOIN BirdSpecies ON BGID = BirdGroupID
WHERE GroupName = 'Thrushes'
ORDER BY SpeciesName ASC;`
- d) `SELECT SpeciesName, count(ObsNo), sum(BirdCount)
FROM BirdSpecies INNER JOIN Observation ON BSID = SpeciesID
WHERE ObsYear = 2015
GROUP BY SpeciesName
ORDER BY sum(BirdCount) DESC;`
- e) `SELECT SpeciesName
FROM BirdSpecies
WHERE BSID IN (SELECT SpeciesID
FROM Observation INNER JOIN Location
ON LocationNo = LNo INNER JOIN
Municipality ON MunicipalityNo = Mno
WHERE MunName = 'Trondheim')
AND BSID NOT IN (SELECT SpeciesID
FROM Observation INNER JOIN Location
ON LocationNo = LNo INNER JOIN
Municipality ON MunicipalityNo = Mno
WHERE MunName = 'Trondheim'
AND ObsYear = 2015);`

Oppgave 3 – Teori (20 %)

- a) $F = \{\text{SpeciesID} \rightarrow \text{SpeciesName}, \text{BirdGroup}, \text{Status}; \text{SpeciesName} \rightarrow \text{SpeciesID}\}$

Forutsetninger:

- SpeciesId er en unik identifikator for en fugleart.
 - En fugleart tilhører bare en fuglegruppe (BirdGroup) og har bare en forekomststatus (Status).
 - Det er ikke flere fuglearter med samme navn (SpeciesName).
- b) Siden attributtene A og D ikke er på noen høyreside i F, må de være med i alle kandidatnøkler. Bare AD er en nøkkel ($AD^+ = ABCDE = R$, $A^+ = ABC \subset R$ og $D^+ = D \subset R$) og det er derfor den eneste kandidatnøkkelen for tabellen.
- c) Ikke-nøkkel-attributtene B og C er delvis avhengige av nøkkelen AD, siden $A \rightarrow B$ og $A \rightarrow C$ (kan utledes). Tabellen oppfyller derfor ikke kravene til 2. Normalform. Den høyeste normalformen som oppfylles av tabellen er derfor 1. normalform.
- d) En mulig dekomponering er: $R_1(A, B)$, $R_2(B, C)$ og $R_3(C, D, E)$. A er primærnøkkel i R_1 . B er primærnøkkel i R_2 . CD er primærnøkkel i R_3 . Dekomponeringer vurderes etter fire forhold:
- **Attributtbevaring:** $R = R_1 \cup R_2 \cup R_3$ så det er ivaretatt.
 - **FD-bevaring:** $A \rightarrow B$ ivaretas i R_1 , $B \rightarrow C$ ivaretas i R_2 og $CD \rightarrow E$ 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 B er (super-)nøkkel i R_2 . Resultatet joiner ikke tapsløst med R_3 til utgangspunktet (R) siden det felles attributtet (C) ikke er en (super-)nøkkel i noen av operandtabellene. Vi har med andre ord ikke oppfylt kravene til tapsløst-join-egenskapen. Egenskapen kan alternativt vises ved å bruke matrisemetoden (algoritme 15.3 i læreboken).
 - **Normalform:** R_1 har $F_1 = \{A \rightarrow B\}$ og er på BCNF siden A er en (super-)nøkkel i R_1 . R_2 har $F_2 = \{B \rightarrow C\}$ og er på BCNF siden B er en (super-)nøkkel i R_2 . R_3 har $F_3 = \{CD \rightarrow E\}$ og er på BCNF siden CD er (super-)nøkkel i R_3 .

Dekomponeringen kan *ikke* brukes siden den ikke har tapsløst-join-egenskapen.

- e) Tabellen trenger bare å inneholde de fire oppgitte tuplene (radene).

Lecturer	Subject	Campus
svein	db	gløs
rune	db	gjøvik
tore	db	kalvskinnet
mads	os	gløs

Oppgave 4 – Lagring og skalering (5 %)

Det er to sentrale opplysninger i oppgaven:

1. Antall poster er usikkert og kan bli stort.
2. All aksess skjer via primærnøkler

Hashing er bra når aksess er via primærnøkler. **Extendible hashing** er bra når datamengden er usikker. Clustered, extendible hashing høres bra ut når du bare skal ha tak i enkeltposter. B+-trær kan også være brukbare, men ikke så bra alternativ her.

Oppgave 5 – B+-trær og queryutføring (15 %)

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

i) `INSERT INTO Ansatt VALUES`

`(123123,'Hansen','Hans','hans@email.org',2015,100000);`

Løsningsforslag: 3 blokker leses på vei nedover B+-treet. 1 blokk skrives etter innsetting. **Til sammen 4.** Ved blokksplitt (hver 30ende innsetting gir blokksplitt og litt flere aksesser).

ii) `SELECT lastname, firstname, email, startyear,salary FROM Employee WHERE empno=12123;`

Løsningsforslag: 3 blokker på vei ned B+-treet.

iii) `SELECT * FROM Employee;`

Løsningsforslag: 3 blokker på vei nedover og 1499 bortover, **til sammen 1502.**

iv) `SELECT COUNT(*) FROM EMPLOYEE WHERE empno>100000.`

Løsningsforslag: Her kan vi navigere ned til empno 100000 og så gå bortover til stigende nøkler i B+-treet. Det er altså 3 aksesser nedover. Hvis vi da sier at andelen av blokker er $(123123-100000)/(123123-100000) = 0.2044$. Når dette ganges med antall blokker på løvnivå, dvs. 1500 får vi 307 blokker. Til sammen ca **309 blokker.**

- b) (5 %) Her er det fristende å anta at stigende empno gir stigende startyear og at vi derfor kan få til optimal aksess via det eksisterende B+-treet, ved å starte på slutten og gå baklengs inntil $\text{startyear} < 2015$. Men det er en skummel antagelse.

Startyear er neppe selektivt nok til at en separat indeks vil lønne seg, derfor sier vi **nei, bruk den eksisterende lagringen.**

Noen studenter har foreslått å lage en sammensatt nøkkel (startyear, empno). Kanskje ikke så dumt?

Oppgave 6 – Transaksjoner - gjenopprettbarhet (10 %)

H1: r1(A); w1(A); r2(B); c1; r2(A); w2(A); c2; H1 er **strict** fordi det er ingen lesing eller skrivning av ikke-committede endringer.

H2: r1(A); w1(A); w2(A); c2; c1; H2 er **ACA** fordi det er ingen lesing av ikke-committede endringer, men det er derimot en skrivning w2(A) skriver over w1(A). Derfor er den ikke strict.

H3: r1(A); r2(C); r1(C); r3(A); r3(B); w1(A); w3(B); r2(B); w2(C); w2(B); c1; c2; c3; H3 er **ikke-recoverable** fordi r2(B) leser endringen til w3(B) og de committer i motsatt rekkefølge.

Oppgave 7 – Transaksjoner - recovery (10 %)

a) Løsning: Vi går forover i loggen og oppdaterer de to tabellene. Da får vi transtabellen:

Transaction	Last_lsn	Status
T1	103	Commit
T2	108	Commit
T3	106	In progress

Og vi får denne DPTen:

Page_id	Rec_lsn
C	101
B	102
A	106

Det er viktig at Rec_lsn **ikke** oppdateres av nye loggposter, dvs. at 101 erstattes med 107.

b) Hvilke loggposter vil legges til loggen under **undofasen** av recovery her?

LSN	Last_lsn	Transaction	OpType	Page_id	Other_info
101	0	T1	Update	C	...
102	0	T2	Update	B	...
103	101	T1	Commit		...
104			Begin_ckpt		
105			End_ckpt		
106	0	T3	Update	A	...
107	102	T2	Update	C	...
108	107	T2	Commit		...
109	106	T3	CLR	A	
110	109	T3	Abort		

Nye loggposter er vist med gult. Det er viktig at CLRen (compensating log record) kommer med her. Noen har skrevet kun en update-loggpost i stedet for CLR. Det må også godtas. Abort kan også komme før CLRen hvis systemet logger at den begynner å abortere.