NTNU
Det skapende universitet

Skisse til løsning av eksamensoppgave i TDT4145 Datamodellering og databasesystemer

Vers: 17.aug 2016

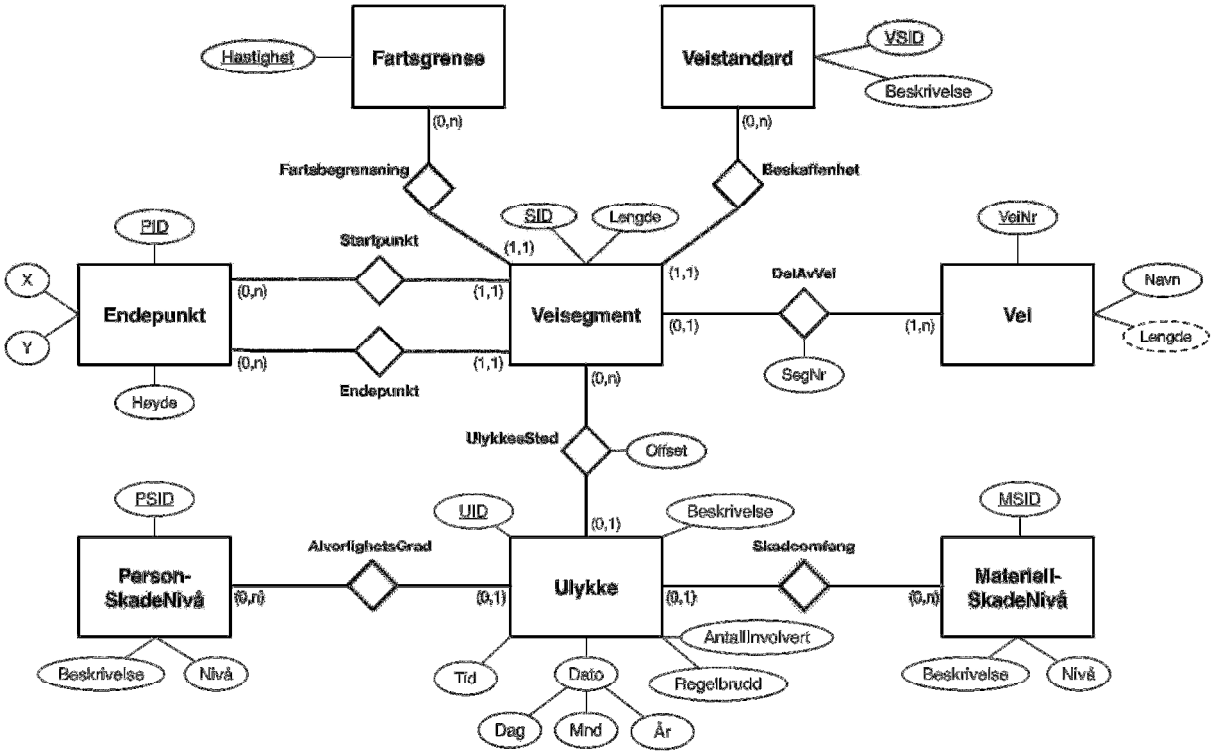
Faglig kontakt under eksamen:
Roger Midtstraum: 995 72 420
Svein Erik Bratsberg: 995 39 963

Eksamensdato: 17. aug 2016
Eksamenstid (fra-til): 15:00 - 19:00
Hjelpemiddelkode/Tillatte hjelpemidler:
D – Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Annen informasjon:

Målform/språk: Norsk bokmål
Antall sider: 7
Antall sider vedlegg: 0

Oppgave 1



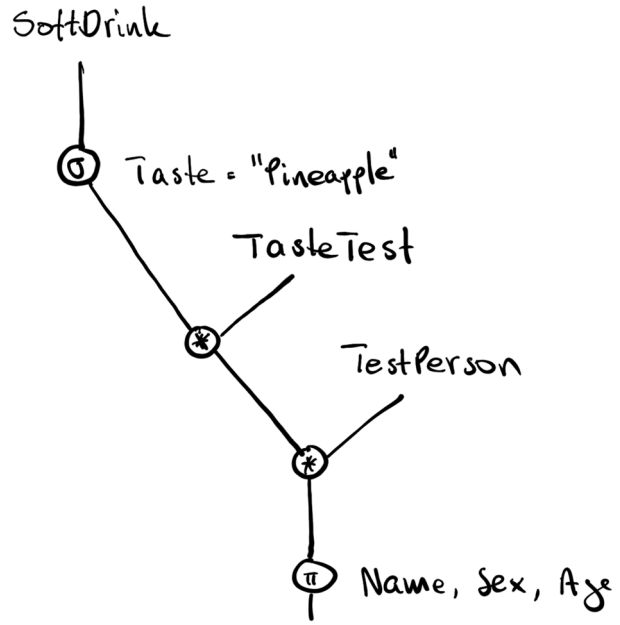
Det er en mulighet å lagre fartsgrense som attributt på Veisegment, men den valgte løsningen anses som bedre med hensyn til til å kunne administrere et sett av standardiserte fartsgrenser. En alternativ modelleringsvalg er å samle PersonSkadeNivå og MateriellSkadeNivå under en felles superklasse, SkadeNivå, som håndterer klassenes felles attributter. En slik løsning skal vurderes likeverdig med løsningen over.

Oppgave 2

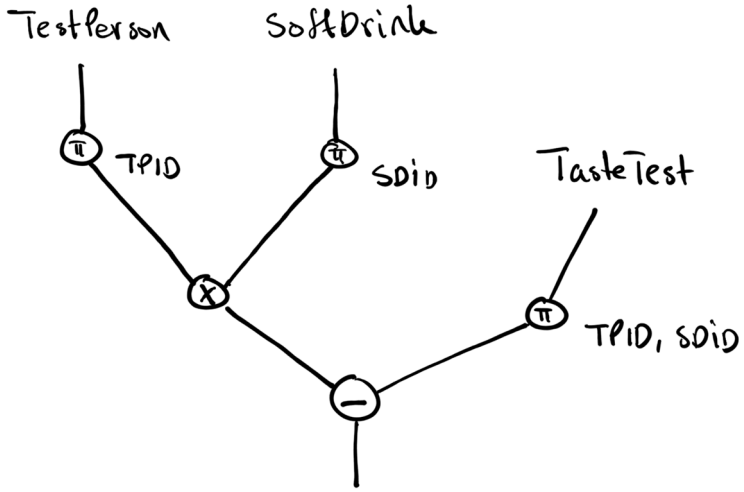
a) Alternativ 1 vil begrense en testpersons mulighet til å smaksteste en brus flere ganger og gi den samme vurdering. Det er ingenting i det oppgitte relasjonsskjemaet som begrenser en slik mulighet. Dersom vi oversetter alternativ 1 til en ekvivalent relasjonsdatabase, vil tabellen få en annen restriksjon enn tabellen i oppgaveteksten, siden kombinasjonen av TPID, SDID og RID vil bli en nøkkel for tabellen.

   Alternativ 2 har ingen av disse problemene og samsvarer derfor bedre med det oppgitte relasjonsskjemaet.

b) Relasjonsalgebra:



c) Relasjonsalgebra:



d) SQL

```
select avg(Age)
from TestPerson
where TPID in (select TPID
               from TasteTest natural join SoftDrink
                   natural join Ranking
               where Taste = ”Pinapple”
                 and Description = ”awesome”)
```

Det er likeverdig å bruke ”inner join” i den indre spørringen. Det skal heller ikke trekkes for å løse oppgaven med join, uten nøsting.

e) SQL:

```
select RID, Description, count(*) as ”Antall”
from TasteTest inner join Ranking
    on (TasteTest.RID = Ranking.RID)
group by RID, Description
order by RID ASC
```

Det er likeverdig å bruke ”natural join” i spørringen. Det er ikke nødvendig å bruke alias for count(*).

f) SQL:

```
select SDID, ProductName,  count(*) as ”Antall”
from SoftDrink natural join TasteTest natural join Ranking
    natural join TestPerson
where Taste = ”Orange”
  and Description = ”awesome”
  and Sex = ”female”
group by SDID, ProductName
having Antall > 5
```

Det er likeverdig å bruke ”inner join” i spørringen. Det er ikke nødvendig å bruke alias for count(*). Oppgaven kan også løses med en nøstet spørring.

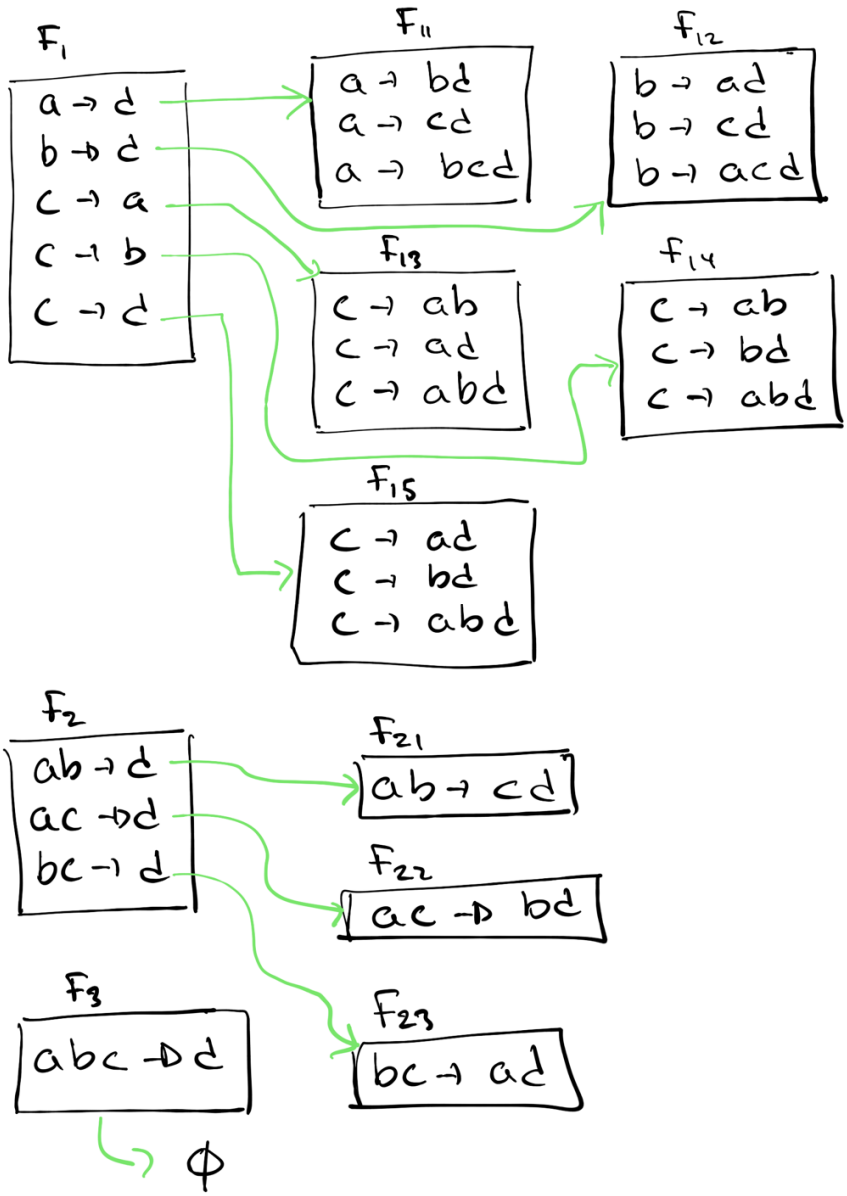
Oppgave 3

a) Når en spesialisering er *disjunkt* kan ikke entiteter være med i flere av subklassene.
b) Når en spesialisering er *total* må alle entiteter være med i minst en subklasse.
c) Dersom det finnes fremmednøkler må disse referere til tupler som finnes eller ha NULL-verdi for alle attributtene i fremmednøkkelen.

Oppgave 4

Vi kan i utgangspunktet se bort fra funksjonelle avhengigheter (FD) med ett eller flere attributter som er med i både venstre og høyre side i den funksjonelle avhengigheten. Om slike FD-er gjelder eller ikke avgjøres av en FD med samme venstreside-attributter, der de overlappende attributtene er fjernet fra høyresiden.

I figuren under har vi vist en løsning der vi tar utgangspunkt i FD-er med henholdsvis ett (F_1), to (F_2) eller tre (F_3) venstreside-attributter som ikke kan gjelde med utgangspunkt i den oppgitte tabellforekomsten.



For hver FD som vi kan utelukke, kan vi i tillegg utelukke alle FD-er som kan utlede den aktuelle FD-en. Et eksempel fra F_1: a->d kan utelukkes. Det betyr at vi heller ikke kan ha a->bd siden vi fra denne kan utlede a->b og a->d. I figuren har vi vist alle "ekstra-FD-er" som kan utelukkes med utgangspunkt i de tre startmengdene.

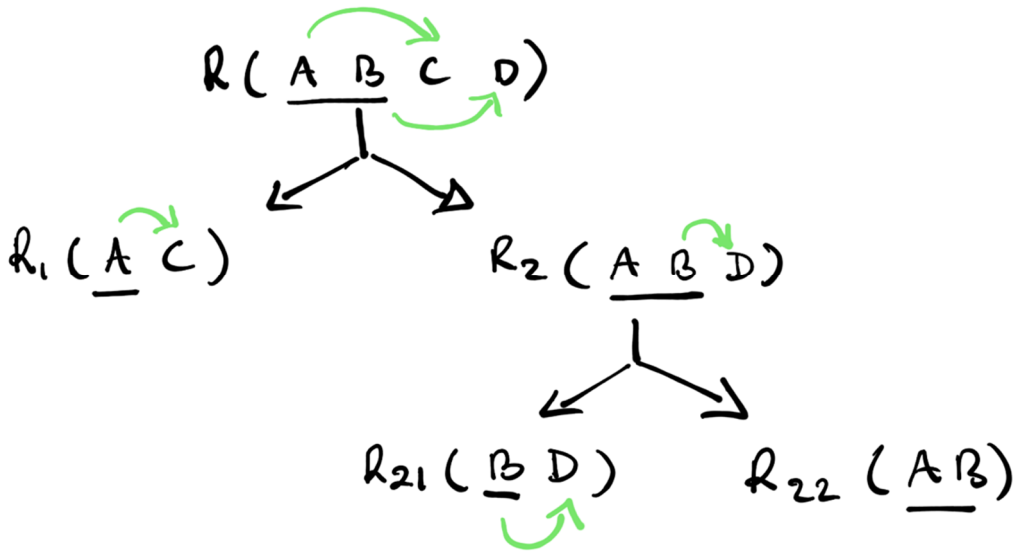
Svaret på oppgaven er: $F_1 \cup F_{11} \cup F_{12} \cup F_{13} \cup F_{14} \cup F_{15} \cup F_2 \cup F_{21} \cup F_{22} \cup F_{23} \cup F_3$

Merknad: Det viste seg at denne oppgaven var mye vanskeligere enn vi opprinnelig tenkte. Ved sensur ble det gitt full uttelling for svar som inneholdt minst F_1. Oppgavetypen er lite aktuell ved fremtidige eksamener i emnet.

Oppgave 5

a) Andre normalform (2NF) tillater ikke at ikke-nøkkel-attributter er avhengige av en ekte delmengde av en nøkkel for tabellen. I R er AB eneste nøkkel. På grunn av A->C og B->D har vi delvise funksjonelle avhengigheter som bryter kravene til 2NF.

b) En god løsning er dekomponering i R_1, R_{21} og R_{22} som vist i under:



- Alle tabellene er på BCNF siden alle funksjonelle avhengigheter har venstesider som er supernøkkel for den aktuelle tabellen.
- Alle attributtene er med i minst en av tabellene.
- Alle funksjonelle avhengigheter er representert i minst en av tabellene.
- R_{21} og R_{22} joiner tapsløst til R_2 siden det felles attributtet (B) er en supernøkkel for R_{21}. R_2 har tapsløst join med R_1 siden det felles attributtet (A) er en supernøkkel i R_1. Siden alle del-dekomponeringene har tapsløst-join-egenskapen, vil dekomponeringen som helhet ha det.

c) Et av mange mulige eksempler som viser at dekomponeringen ikke har tapsløst-join-egenskapen:

Utgangspunkt: R(A B C D)

A	B	C	D
1	1	1	1
2	1	2	2
3	2	1	1

Vi får da: R_1(A B) og R_2(B C D)

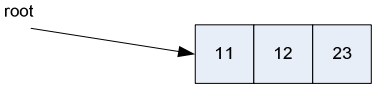
A	B
1	1
2	1
3	2

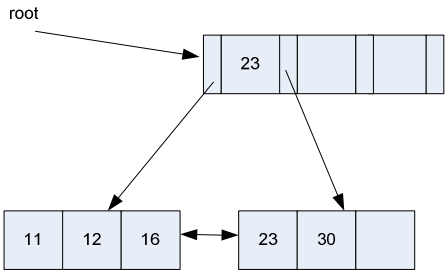
B	C	D
1	1	1
1	2	2
2	1	1

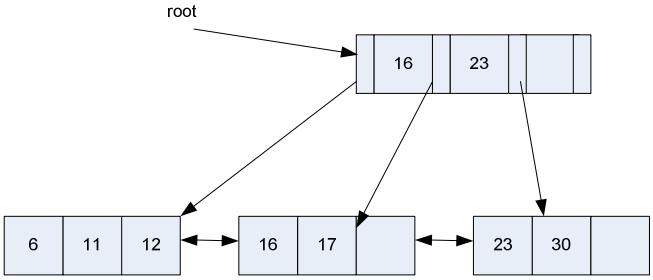
Joiner R_1 og R_2 : $R_1 * R_2$ (A B C D)

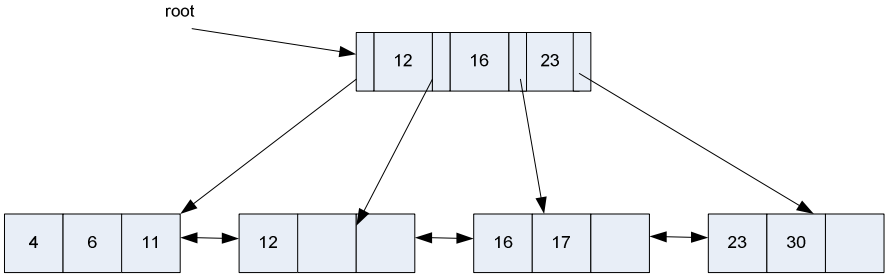
A	B	C	D	
1	1	1	1	ok
1	1	2	2	X
2	1	1	1	X
2	1	2	2	ok
3	2	1	1	ok

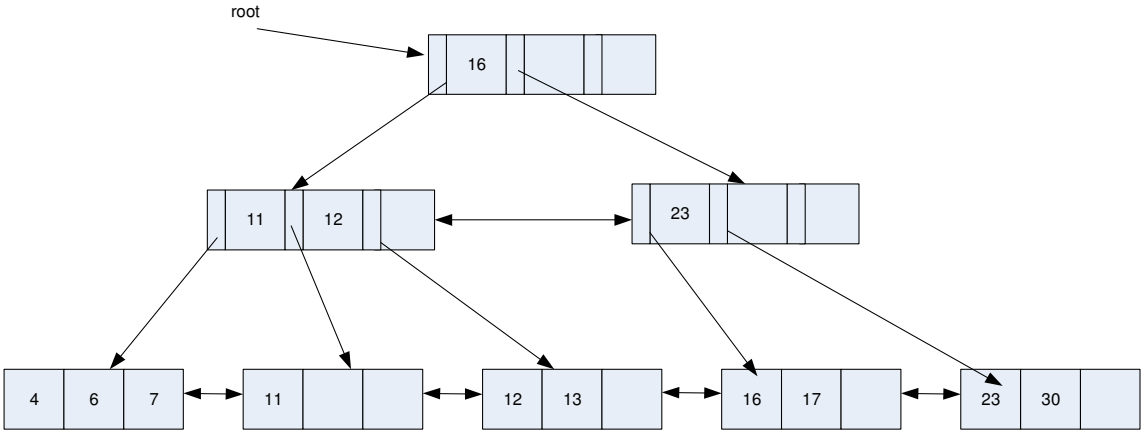
Oppgave 6


root
11
12
23


root
23
11
12
16
23
30


root
16
23
6
11
12
16
17
23
30


root
12
16
23
4
6
11
12
16
17
23
30


root
16
11
12
23
4
6
7
11
12
13
16
17
23
30

Oppgave 7

a) Her er det lurt med en clustered hash på staffNo fordi det er oppdatering av ansatte via dette attributtet. I tillegg kan det være lurt med en unclustered B+-tre-index på salary, slik at det er lett å finne den/de ansatte med høyest lønn. Et clustered B+-tre på staffNo kan også erstatte hash-strukturen.

b) Vi har 7 plasser i buffer. Bruker 5 av de til Department, 1 til Staff og 1 til joinresultat. Vi får da lest Department i biter seks ganger og hver gang må vi lese Staff. Dvs. vi får 6*(5 + 1000) = 6030.

Oppgave 8

a) Ikke-serialiserbar: 1 -> 2 -> 3 og 1 <-> 3.

b)

T1	T2	T3
----	----	----
rl1(X)		
r1(X)		
	rl2(Y)	
	r2(Y)	
rl1(Y)		
r1(Y)		
		rl3(X)
		r3(X)
		rl3(Y)
		r3(Y)
prøver wl1(X)		
må vente		
		prøver wl(Y)
		må vente
	rl2(X)	
	r2(X)	
	wl2(Z)	
	w2(Z)	
	C2; slipper Y,X,Z-låser	

Vranglås mellom T1 og T3. En av de må aborteres og låsene slippes slik at den andre fullfører. Så kan den som ble abortert restartes.

Oppgave 9

a) Vi ser på en loggpost med LSN og BlockId / PageId.
1. Hvis PageId ikke finnes i DPT, kan vi slippe REDO
2. Hvis Loggpostens LSN er mindre enn DPTs tilsvarende RecoveryLSN, kan vi slippe REDO.

3. Hvis Loggpostens LSN er mindre eller lik Blokkas/Pagens PageLSN, kan vi slippe REDO.

b) Det som skjer ved commit er at transaksjonens loggposter skrives ut til disk. Evt. skitne datablokker i buffer trenger ikke å skrives ut her. De kan skrives ut senere.