NTNU
Det skapende universitet

Skisse til løsning av eksamensoppgave i TDT4145 Datamodellering og databasesystemer

Vers. 0.3, 20.juni 2016

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

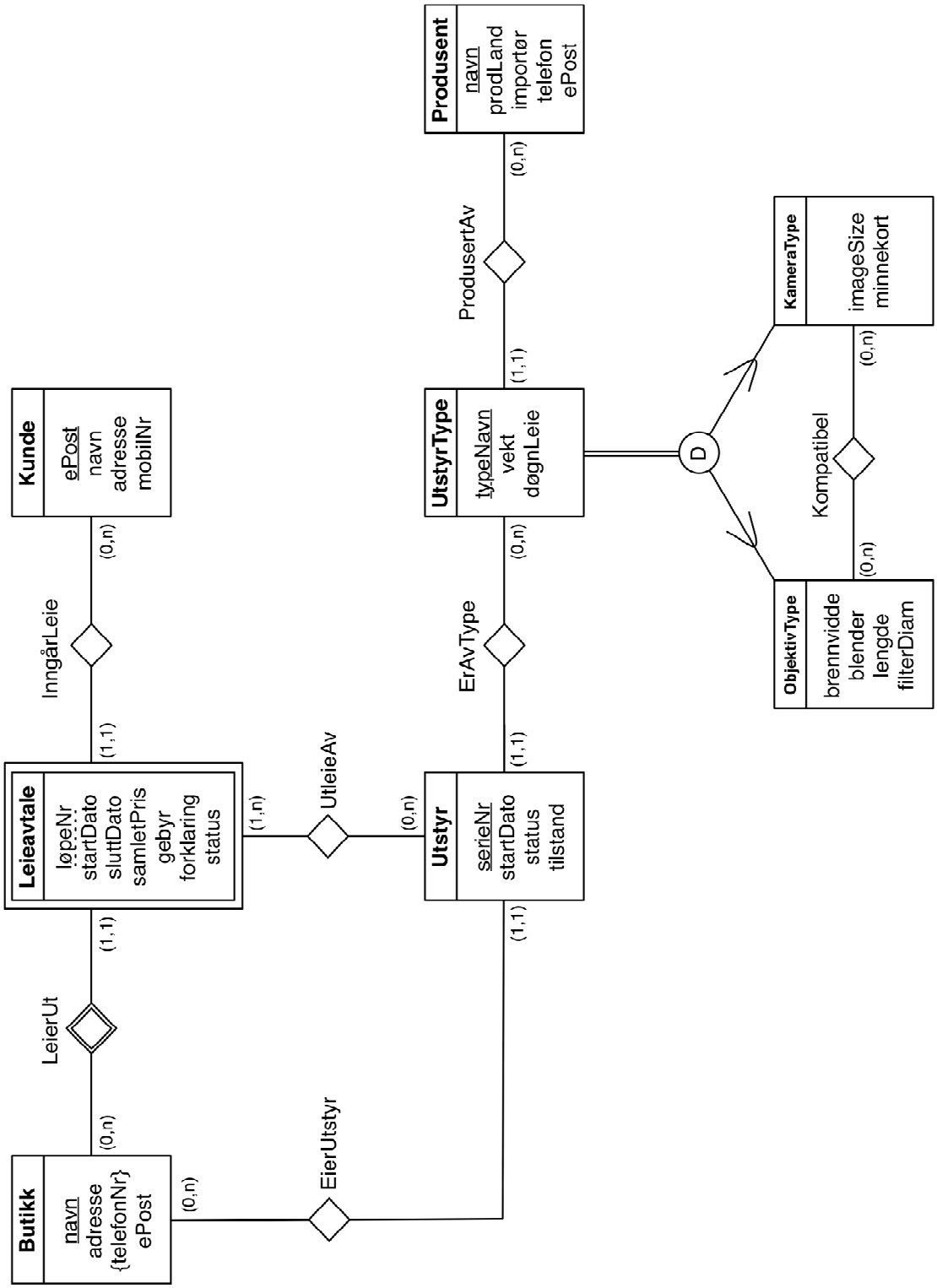
Eksamensdato: 4. juni 2016
Eksamenstid (fra-til): 09:00 - 13:00
Hjelpemiddelkode/Tillatte hjelpemidler:
D – Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Annen informasjon:

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

Oppgave 1

Vi har valgt å modellere LeieAvtale som en svak entitetsklasse, som identifiseres via



LeierUt-relasjonsklassen mot Butikk. Dette forutsetter at hver butikk vedlikeholde en serie av unike løpeNr for sine utleier.

Oppgave 2

a) I figuren under har vi vist resultatet av henholdsvis naturlig join, full-outer-join og kartesisk produkt (cross join).

A	B	C	D
1	1	2	4
3	3	6	4

A	B	C	B	D
1	1	2	1	4
2	2	4	null	null
3	3	6	3	4
null	null	null	5	4

A	B	C	B	D
1	1	2	1	4
1	1	2	3	4
1	1	2	5	4
2	2	4	1	4
2	2	4	3	4
2	2	4	5	4
3	3	6	1	4
3	3	6	3	4
3	3	6	5	4

b) Ut over å spesifisere hvilken tabell fremmednøkkelen refererer til og om den kan ta nullverdi eller ikke, kan man spesifisere hva som skal skje dersom det skjer endring i eller sletting av det tuppelet fremmednøkkelen refererer til.

Aktuelle handlinger kan være å blokkere endring eller sletting i den refererte tabellen, å videreføre endringen ved enten å endre fremmednøkkelens verdi eller å slette tuppelet med fremmednøkkelen, eller å sette fremmednøkkelen til NULL.

Under er vist syntaks fra MySQL, men det er ikke nødvendig å svare med konkret SQL-syntaks.

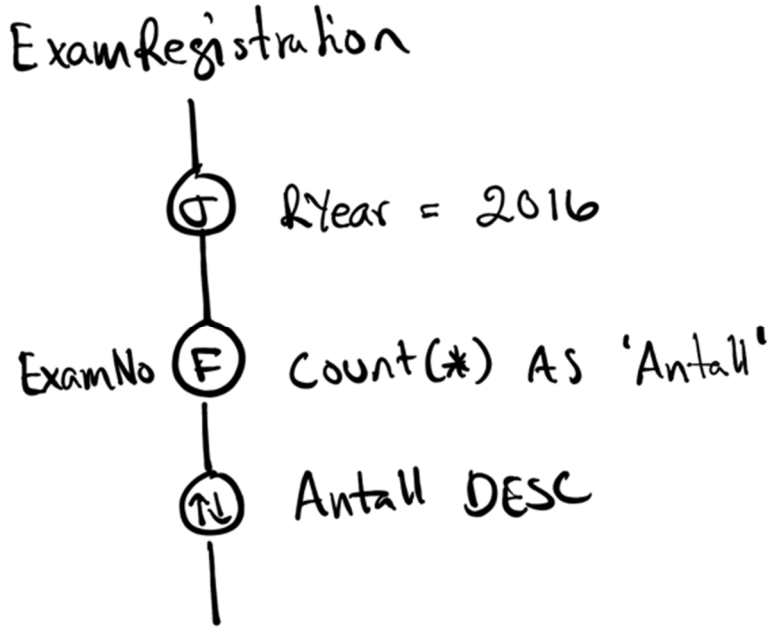
```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name,...)
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

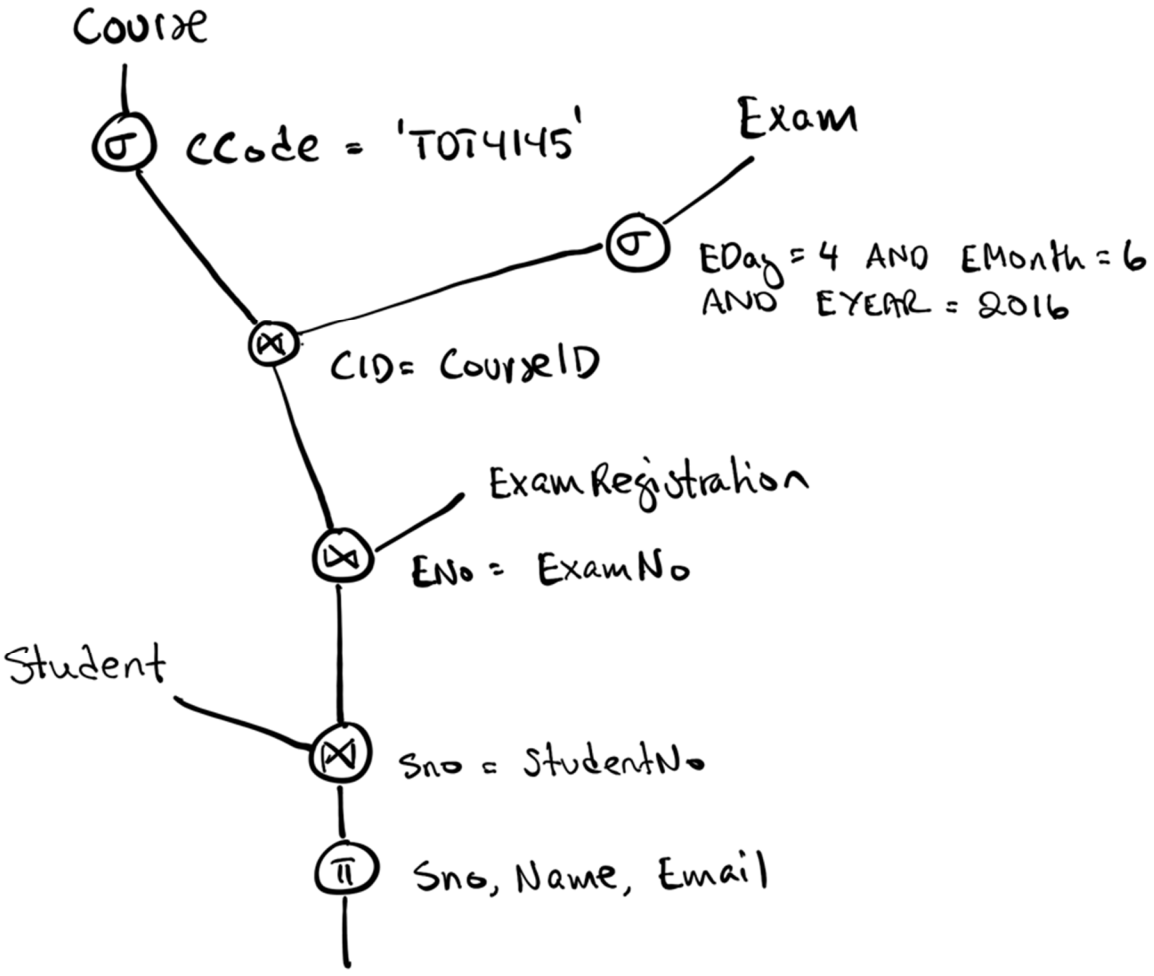
c) *Entitetsidentitet* handler om å sikre at alle rader i en tabell er ulike og realiseres ved at tabellen må ha en primærnøkkel som ikke kan inneholde NULL-verdier.

Oppgave 3

a) Relasjonsalgebra. Det skal ikke trekkes for løsninger uten sortering:



b) Relasjonsalgebra:



c) SQL:

```
SELECT SNo, Name, Email
FROM Student WHERE SNo NOT IN (SELECT StudentNo
                 FROM ExamRegistration INNER JOIN Exam
                      ON (ExamNo = Eno)
                 WHERE EYear = 2015);
```

d) SQL:

```
SELECT Grade, Count(*)
FROM Course INNER JOIN Exam ON (CID = CourseID)
            INNER JOIN ExamResult ON (Eno = ExamNo)
WHERE CCode = 'TDT41451'
  AND EDay = 1 AND EMonth = 6 AND EYear = 2015
GROUP BY Grade
ORDER BY Grade ASC;
```

e) SQL:

```
INSERT INTO Exam
VALUES (999, 100, 4, 6, 2016, 4, 'Written');
```

Man kan også skrive

```
INSERT INTO Exam (ENo, CourseID, EDay, EMonth, Duration, EType) VALUES
(999, 100, 4, 6, 2016, 4, 'Written');
```

Oppgave 4

a) Funksjonelle avhengigheter: AthleteID -> Name, Club, Class
AthleteID, Trial -> Length, Wind, Status

Den første funksjonelle avhengigheten forutsetter at AthleteID er en unik identifikator for utøver og at en utøver har samme verdi for Name, Club og Class i hele tabellen.

Den andre funksjonelle avhengigheten forutsetter at kombinasjonen av AthleteID og Trial er en unik identifikator for et hopp (en rad i tabellen).

b) Gitt funksjonelle avhengigheter som foreslått over, vil kombinasjonen av AthleteID og Trial være den eneste minimale supernøkkelen i tabellen, og dermed tabellens (kandidat-)nøkkel.

c) Det som er uheldig med denne tabellen er at utøverinformasjon gjentas for hvert hopp som utøveren gjør. Attributtene Name, Club og Class er delvis avhengig av nøkkelen. Dette forhindrer at tabellen oppfyller andre normalform.

Problemer vil være innsettings-, slettings- og oppdateringsanomalier. Vi kan ikke registrere utøverinformasjon før utøveren har minst ett hopp. Ved sletting av utøverens hopp vil utøverinformasjonen gå tapt. På grunn av redundant lagring av utøverinformasjonen vil man ved endring kunne måtte oppdatere i flere rader og inkonsistens kan oppstå mellom rader som gjelder samme utøver.

Oppgave 5

a) Vi kan legge til A -> C eller B -> C. I begge tilfeller vil vi få $A^+ = ABCD = R$ som er et kriterium for at A er en supernøkkel.

b) A og B er ikke med på høyresiden i noen funksjonelle avhengigheter og må derfor være med i alle supernøkler. Siden $AB^+ = ABCD = R$ ser vi at AB er en supernøkkel. Verken A ($A^+ = A$) eller B ($B^+ = B$) er supernøkler. Dermed kan vi slutte at AB er tabellens eneste nøkkel. Det gir oss A og B som nøkkelattributter, og C og D som ikke-nøkkelattributter.

Vi har ingen delvis avhengigheter av nøkkelen ($A^+ = A$ og $B^+ = B$) og kan derfor slutte at tabellen er på andre normalform.

For at tabellen skal være på tredje normalform (3NF) må det for alle funksjonelle avhengigheter på formen X -> Y, være slik at X er en supernøkkel eller Y er et nøkkelattributt.

AB -> C : AB er en supernøkkel, altså ok i forhold til 3NF

BC -> D : BC er ikke en supernøkkel og D er ikke et nøkkelattributt, bryter 3NF

På grunn av den funksjonelle avhengigheten BC -> D er tabellen ikke på tredje normalform.

c) R_1 og R_2 inneholder alle attributter i R, altså har vi attributtbevaring.

$F_1 = \{AB->C\}$ og $F_2 = \{BC->D\}$. Siden $F_1 \cup F_2 = F$ har vi bevaring av funksjonelle avhengigheter.

I R_1 har vi den funksjonelle avhengigheten AB->C. Siden AB er en supernøkkel for tabellen oppfylles kravet til Boyce-Codd normalform (BCNF).

I R_2 har vi den funksjonelle avhengigheten BC->D. Siden BC er en supernøkkel for tabellen oppfylles kravet til Boyce-Codd normalform (BCNF).

R_1 og R_2 har BC som felles attributter. Siden BC er en (super-)nøkkel i R_2 har dekomponeringen tapsløst-join-egenskapen.

Dekomponeringen oppfyller alle fire kravene til dekomponeringer og er derfor av god kvalitet.

Oppgave 6

En post i Hashfila vil være slik (ssn, fname, lname, super_ssn, dno). En post på løvnivå i B+-treet vil være slik (lname, ssn).

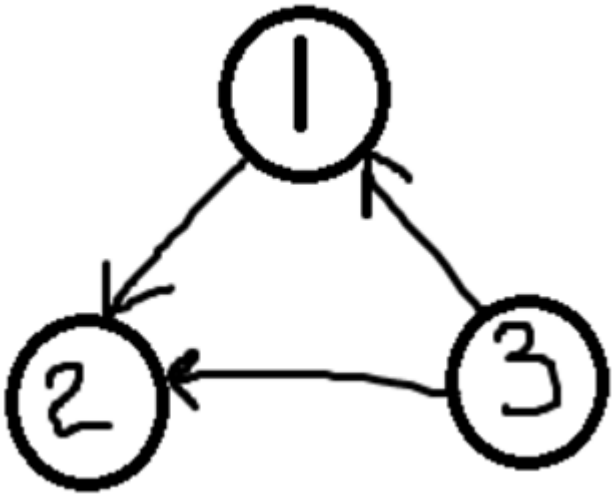
i) 1.2 (les hash, gj.snitt) + 3 (les nedover B+-tre) + 1 (skriv hash) + 1 (skriv B+-tre) = **6.2** (gj.snitt). Dette er antall I/Oer. Antall aksesser kan også tolkes til antall lesinger: Da er riktig gj.snitt svar **4.2**.
ii) **1.2** (les hash, gj.snitt)
iii) **3** (les nedover B+-tre) + **N*1.2** (les hash), N er antall som heter Hansen. Hvis antall Hansen blir stort, må også flere B+-tre-blokker leses.
iv) Kan ikke bruke B+-tre til å søke etter salary. Scan hashfil: **1250** blokker.
v) Index-only query: les 3 nedover B+-tre + 599 bortover B+-tre = **602**.

Oppgave 7

i) **Unrecoverable**. T2 leser dirty X fra T3 og T2 committer før T3.
ii) **Recoverable**. T3 leser dirty Y fra T2, men T2 committer før T3.
iii) **Strict**. Ingen lesing eller skriving av ikke-committede endringer.

Oppgave 8

a)



Konfliktserialiserbar

b) rl = read_lock, wl = write_lock. Tida går nedover i denne figuren:

T1	**T2**	**T3**
rl1(X); r1(X);		
	rl2(Z); r2(Z);	
rl1(Z); r1(Z);		
		rl3(X); r3(X);
		rl3(Y); r3(Y);
(w1(X); kan ikke gjøre denne, da T1 må vente på lås på X);		
		wl3(Y); (upgrade)
		w3(Y);
		c3; (slipper låser for X og Y);
(T1 vekkes opp)		
wl1(X); w1(X);		
c1; (slipper låser for X og Z)		
	rl2(X); r2(X);	
	wl2(Z); w2(Z);	
	wl2(Y); w2(Y);	
	c2; (slipper låser for X, Y, Z)	

Oppgave 9

a) Transaksjonstabell

Trans	**Last_LSN**	**Status**
T1	106	Active

T2	105	Commited

DPT

DataBlock	**RecoveryLSN**
E	103
C	104

b) Blokk E vil da ha **PageLSN 103** fordi den ikke trenger redo.
Vi vet ikke PageLSN for Blokk C, men den vil ha en PageLSN mindre enn 104, også mindre enn 101 da ingen av 101, 102 eller 103 endrer blokk C. **PageLSN < 101**.