Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

LØSNINGSFORSLAG TIL KONTINUASJONSEKSAMEN I FAG TDT4145 – DATAMODELLERING OG DATABASESYSTEMER

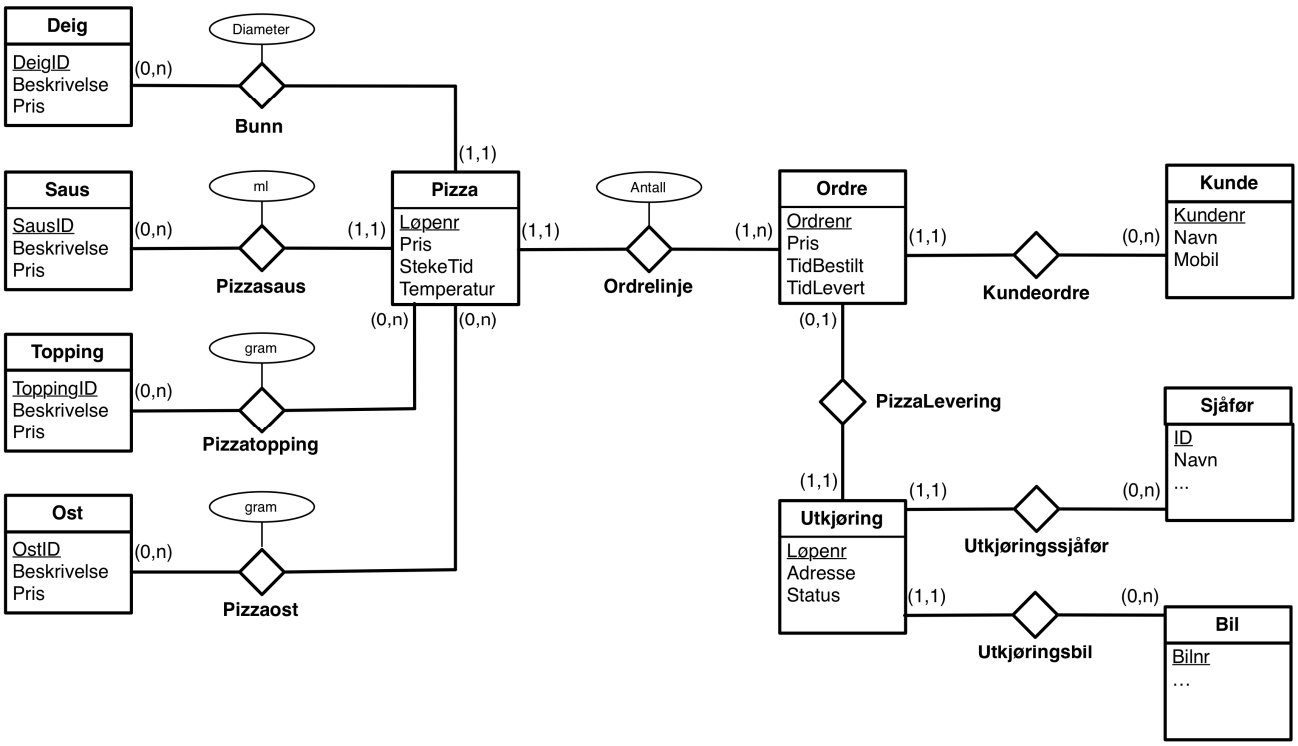
Faglig kontakt under eksamen: Svein Erik Bratsberg

Tlf.: 99539963 (Bratsberg)

Eksamensdato:	**6. august 2012**
Eksamenstid:	**09.00-13.00**
Tillatte hjelpemiddel:	**D: Ingen trykte eller håndskrevne hjelpemiddel tillatt. Bestemt, enkel kalkulator tillatt.**

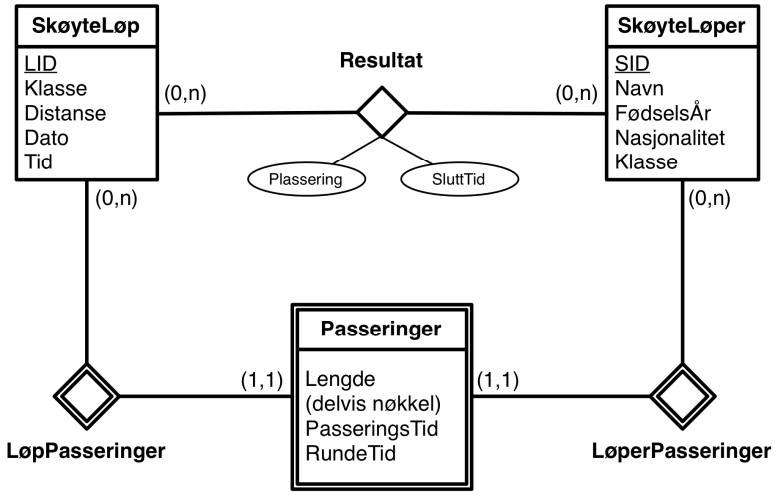
Språkform: Bokmål

Oppgave 1 – Datamodellering – 20 %

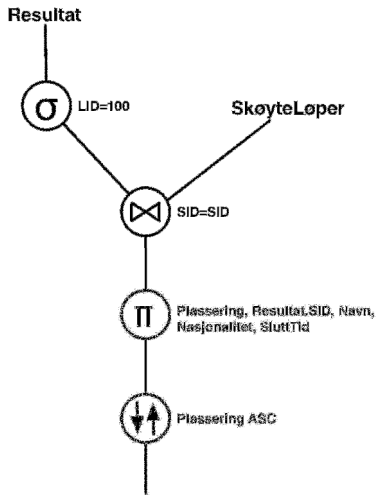


Oppgave 2 – ER, relasjonsalgebra og SQL – 20 %

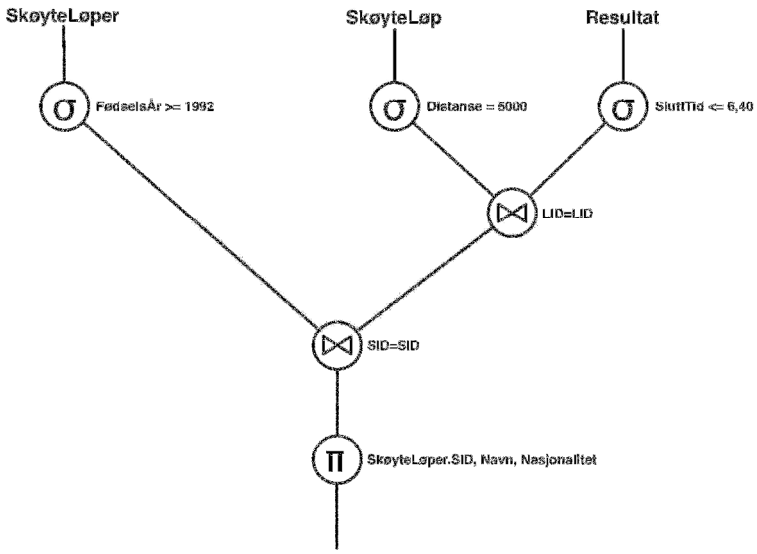
a) En løsning på oppgaven (blant flere mulige):



b) Relasjonsalgbra i grafnotasjon(en løsning uten sortering godtas uten trekk):



c) Relasjonsalgebra i grafnotasjon:



d) Spørring i *SQL* (løsninger med join-predikatet i from-delen er likeverdige):

```
select Plassering, s.SID, Navn, Nasjonalitet, SluttTid
from SkøyteLøper AS s, Resultat AS r
where s.SID = r.SID
  and LID = 100
order by Plassering ASC, Navn ASC
```

e) Spørring i *SQL* (løsninger med join-predikatene i from-delen er likeverdige):

```
select Nasjonalitet, count(r.SID)
from Resultat AS r, SkøyteLøper AS s
where r.SID = s.SID
  and LID = 100
group by Nasjonalitet
order by count(r.SID) DESC
```

Oppgave 3 – Teori – 20 %

a) Et tuppel (a_x, b_1, c_1, d_2, e_x) der $d_2 \neq d_1$ og a_x og e_x er vilkårlige verdier, vil gjøre at BC → D *ikke* kan gjelde for tabellen.

b) To mengder funksjonelle avhengigheter F og G er *ekvivalente* hvis de uttrykker den samme restriksjonen. Det kan uttrykkes som $F^+ = G^+$. For å sjekke om to spesifiserte mengder er ekvivalente, er det lite hensiktsmessig (les: mye arbeid) å beregne tillukningene. Det er enklere å sjekke om FD-ene i den ene mengden kan utledes fra FD-ene i den andre mengden og vise versa.

F utleder G: a→b og b→c gir a→c. a→b og a→c gir a→bc. c→a er med i begge mengdene. F utleder med andre ord G (G^+ er en delmengde av F^+).

G utleder F: a→bc gir a→b. c→a er med i begge mengdene. b→c kan *ikke* utledes fra FD-ene i G (F^+ er ikke en delmengde av G^+).

F og G er *ikke* ekvivalente mengder med funksjonelle avhengigheter.

c) En kandidatnøkkel er en unik identifikator (supernøkkel – aldri to tupler/rader med samme verdi for alle attributtene i identifikatoren) som er *minimal*. At identifikatoren er minimal betyr at det ikke finnes noen ekte delmengde av attributtene i identifikatoren som også er en unik identifikator.

Kandidatnøkler i R: AD, BD og CD.

d) Tredje normalform (3NF) godtar alle funksjonelle avhengigheter der de attributtene som bestemmes (høyresideattributtene) er nøkkelattributter (med i minst en kandidatnøkkel). I R er alle attributtene nøkkelattributter. Tabellen vil derfor være på 3NF uansett hvilke funksjonelle avhengigheter som gjelder.

Boyce-Codd nomalform (BCNF) krever at alle mengder av attributter som bestemmer noe (venstrsidene i de funksjonelle avhengighetene), må være supernøkler for tabellen. I dette tilfellet er verken A, B eller C supernøkler for tabellen. Tabellen oppfyller derfor *ikke* kravene til BCNF.

Oppgave 4 – Hashing – 15 %

a) Extendible hashing er en hashingmetode for dynamiske data, dvs. hvor datamengden kan endre seg mye over tid og kan være uforutsigbar. Den store fordelen med denne metoden er at *ytelsen ikke blir degradert i forhold til å ha lange overløp som i statisk hashing når datamengden vokser*. En annen fordel er at man kun splitter ei og ei blokk av gangen når ei blokk går full. Ulempen er at man må ha en katalogstruktur som må være minnebasert for å være effektiv. Evt. blir det to blokk- og diskaksesser for hver dataaksess.

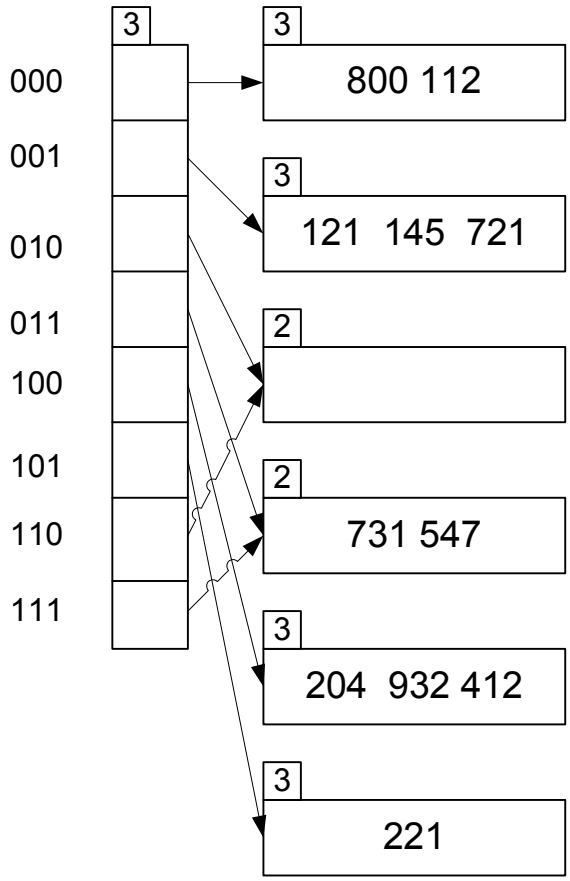
b) Fordelen med lineær hashing er at det er en dynamisk hashingmetode som ikke gjør bruk av katalog, dvs. man har stor sjanse for at man greier seg med en blokkaksess. Ulempen er at man må bruk en overflytsmetode som i statisk hashing da det ikke splittes der hvor behovet nødvendigvis er, men etter et lineært mønster.

Vi mod'er alle verdier på 8:

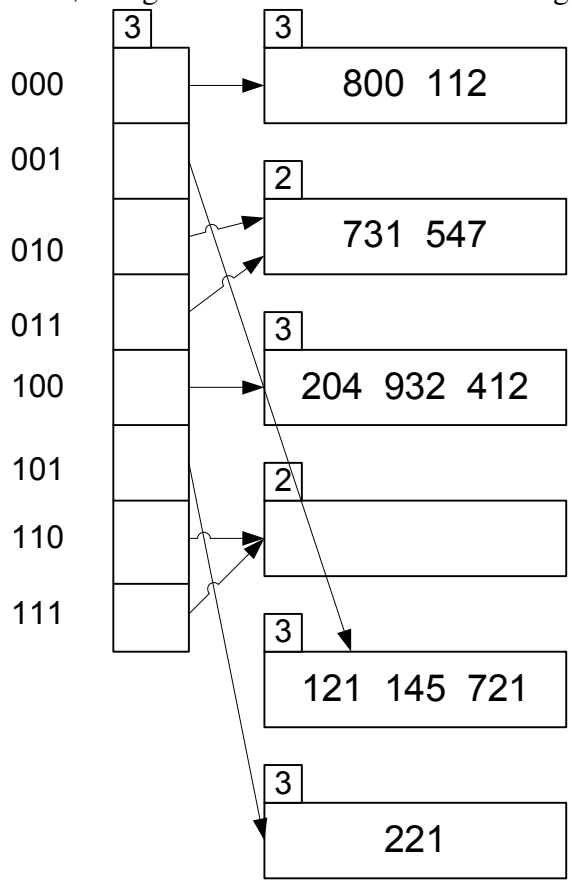
121	1	001
204	4	100
731	3	011
547	3	011

800	0	000
221	5	101
932	4	100
145	1	001
112	0	000
721	1	001
412	4	100

Her viser vi løsningen hvor vi har brukt de minst signifikante bitene av hashverdien (som forelest):



Her har vi vist løsningen når vi bruker de mest signifikante bitene av hashverdien (som i læreboka):



Oppgave 5 – Transaksjonsteori – 10 %

a) **w1(A);r1(B);w2(B);c1;w2(A);c2;**
En strict historie tillater ikke en transaksjon å lese eller skrive en ikke-committet (eller ikke-abortert) verdi.

b) **w1(A);w1(B);w2(A);c1;r2(B);c2;**
En historie er ACA hvis hver transaksjon i historien bare leser verdier skrevet av committede transaksjoner.

c) **w2(A);w1(B);w1(A);r2(B);c1;c2;**
En historie er gjenopprettbar når transaksjoner committer etter transaksjoner de har lest fra committer.

d) **w2(A);w1(B);w1(A);r2(B);c2;a1;**
Her leser T2 B som T1 har skrevet, så commiter T2 og senere borterer T1.

Oppgave 6 – Låsing – 10 %

a)

a. Basic 2PL; Setter og slipper låser når man trenger dem, men må sette alle låser før man slipper noen. Frigjør data tidlig.

   b. Conservative 2PL: Må erklære hvilke data man låser på forhånd. Tillater ikke så mye samtidighet. Men er vranglåsfri.
   c. Strict 2PL: Som basic 2PL, men slipper skrivelåsene helt til slutt. Denne impliserer stricte historier som er basisen for undo-logging.
   d. Rigorous 2PL. Som Strict 2PL, men slipper alle låser til slutt. Enkel å implementere, men tillater mindre samtidighet enn strict 2PL.

b) Vranglås kan oppstå for alle variantene, bortsett fra konservativ 2PL. Dette kan skje ved at to transaksjoner prøver å låse de samme dataene, men gjør det i forskjellig rekkefølge, slik at de blir å vente på hverandres låser i en sykel.

Oppgave 7 – Logging og recovery – 5 %

ARIES lager periodisk sjekkpunktloggposter som brukes til å forkorte tida det tar å gjøre recovery. I en sjekkpunktloggpost (dvs. sluttposten) er et bilde av transaksjonstabellen og «dirty page table» lagret. Transaksjonstabellen brukes til å vite hvilke transaksjoner som er i live ved starten av sjekkpunktet i loggen, slik at man slipper å scanne hele loggen. Dirty Page Table brukes til å finne et sikkert og godt punkt for å starte REDO-recovery, slik at man slipper å starte dette veldig langt tilbake i loggen. ARIES er spesiell her da skriving at skitne blokker er frakoblet sjekkpunktingprosessen.