Eksamen Inf3100 2007

I

Salg(salgsID, kundeID, artsnavn, dato, antall)
Hendelse(hendID, salgsID)
Epikrise(hendID, dato, status, info)

a)
```
create table Salg (
    salgsID  int primary key,
    kundeID  int not null,
    artsnavn varchar(30) not null,
    dato date not null,
    antall int not null,
    unique (kundeID, artsnavn, dato )
);

create table Hendelse (
    hendID  int primary key,
    salgsID int not null references Salg(salgsID)
);
```

int eller noe annet passende

varchar eller char med en passende parameter

Jeg finner det nemlig at alle attributtene skal ha en verdi.

kandidatnøkkel

b)
```
select count(distinct kundeID)
from Salg
where artsnavn = 'Black Molly' and dato >= date '2006-01-01'
                               and dato <= date '2006-12-31';
```

c)
```
select s.salgsID, s.artsnavn, count(h.hendID) as ant
from Salg s, Hendelse h
where s.salgsID = h.salgsID and
      s.dato >= '2006-01-01' and
      s.dato <= '2006-12-31'
group by s.salgsID, s.artsnavn
having count(h.hendID) >2;
```

← Tar med artsnavn for enkelt å kunne få det med i select-klausulen.

Alternativt kan man ta
`group by s.salgsID`
og ha
`select s.salgsID, max(s.artsnavn), count(...)`

I d)

Idé: Tell opp hvor mange hendelser det er for hver salgsID og sammenlikn med antall hendelser for salgsID-en der minst én av epikrisene har status 'diagnose'.

```
select s.salgsID, s.dato
from Salg s
where
      s.dato >= date '2007-01-01' and
      s.dato <= date '2007-01-31' and

      ( select count(*)                                  } antall hendelser
        from Hendelse h1                                   knyttet til dette
        where h1.salgsID = s.salgsID)                      salget (s.salgsID)

      =

      (select count(distinct h.hendID)                   } antall forskjellige
       from Hendelse h2, Epikrise e                        hendelser hvor det er
       where h2.salgsID = s.salgsID and                    minst én epikrise
             h2.hendID = e.hendID and                      med status diagnose
             e.status = 'diagnose')                        (må ha distinct i count
                                                           fordi ellers kan komme
order by s.dato;                                           til å telle en hendID
                                                           flere ganger)
```

(Det er mange måter å besvare denne oppgaven på, f.eks. ved bruk av exists ...)

e) $\sigma_{ant>2}(\gamma_{salgsID, artsnavn, count(hendID) \rightarrow ant}(\sigma_{2006\text{-}01\text{-}01 \leq dato \leq 2006\text{-}12\text{-}31}(Salg \bowtie Hendelse)))$

Inf 3100 2007

II

Adresse(by, postnr, gate)

by, gate → postnr
postnr → by

a) gate forekommer ikke i noen høyreside og må derfor være med i alle kandidatnøkler.

$gate^+ = gate$

$(by, gate)^+ = by, gate, postnr$

$(gate, postnr)^+ = gate, postnr, by$

Kandidatnøklene er derfor (by, gate) og (gate, postnr).

b) I by, gate → postnr er venstresiden en kandidatnøkkel (og derfor supernøkkel), altså er denne FDen på BCNF.

I postnr → by er venstresiden ikke supernøkkel, men høyresiden er et nøkkelattributt, så den er på 3NF, men bryter BCNF.

c)
Adresse(by, postnr, gate)

dekomponeres i henhold til bruddet i postnr → by

R1(postnr, by) R2(postnr, gate)

Fortsatt gjelder FDen by, gate → postnr, men den går på tvers av de to nye relasjonene, så dekomposisjonen er ikke FD-bevarende.

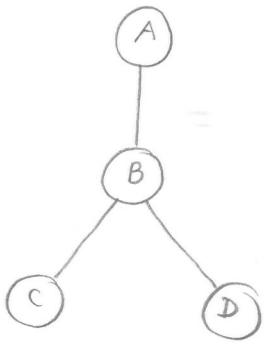
d) I dette tilfellet er det nemlig å anta at det sjelden er oppdateringer av Adresse, mens det ved queries svært ofte ("alltid"?) vil være behov for å joine R1 og R2 for å få ut "hele" adressen. Da kan det lønne seg å beholde tabellen Adresse: ved eventuelle oppdateringer må det foretas en sjekk internt i tabellen for å se at FDene bevares, mens det ved queries kan hentes ut fullstendig adresseinformasjon fra denne ene tabellen.

III

a) Treprotokollen benyttes når den underliggende datastrukturen er et tre (dataelementene som inngår i transaksjonene, er organisert i et tre).

b) Første lås settes på en vilkårlig node i treet,
Deretter kan transaksjonen bare låse videre nedover i treet, på følgende måte: ~~For~~ å ta lås på en node, må man ha lås på foreldrenoden. Låser kan slippes når man vil, men hvis en node har hatt en lås og deretter sluppet den igjen, får den ikke ta låsen på nytt (selv om foreldrenoden fortsatt har sin lås).

c)



T_1: $l_1(A); r_1(A); l_1(B); u_1(A); r_1(B); l_1(C); u_1(B); r_1(C); w_1(C); u_1(C)$

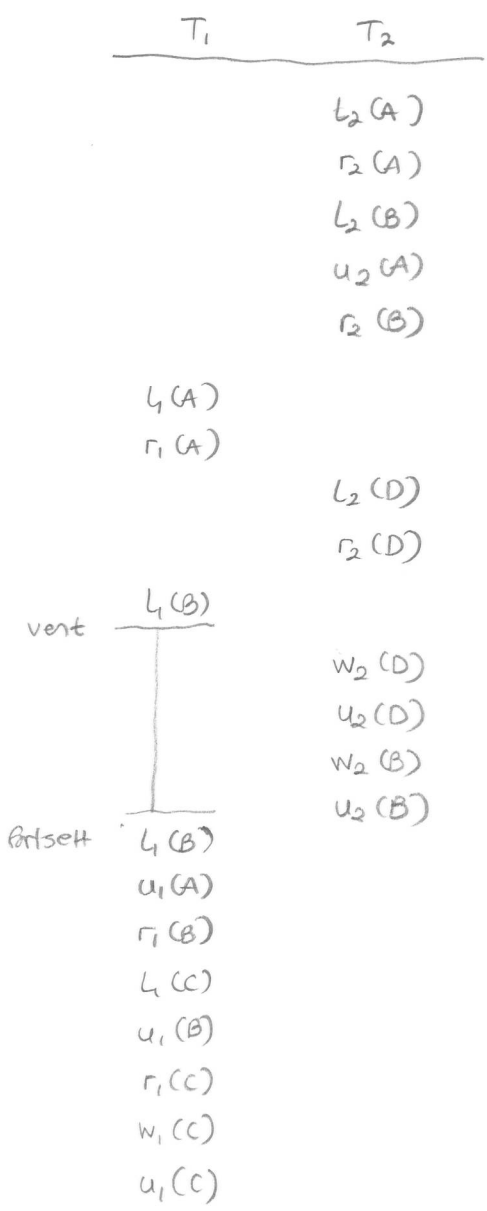
T_2: $l_2(A); r_2(A); l_2(B); u_2(A); r_2(B); l_2(D); r_2(D); w_2(D); u_2(D); w_2(B); u_2(B)$

Planen blir:

$l_2(A); r_2(A); l_2(B); u_2(A); r_2(B); l_1(A); r_1(A); l_2(D); r_2(D); l_1(B); u_1(A); r_1(B); l_1(C); u_1(B); r_1(C); w_2(D); u_2(D); w_1(C); u_1(C); w_2(B); u_2(B)$

Inf 3100 2007

III c) (forts.)



Inf 3100 2007

IV

a) TPMMS: Formålet er sortering når en relasjon er for stor til å få plass i primærminnet.

Fase1: Sorterer så store biter som man kan få plass til i minnet, og skriver hver delsortering til disk, av gangen

Etter fase 1 har vi grupper av blokker som er fullstendig sortert hver for seg, "sublister"

Fase 2: Henter inn en blokk fra hver subliste til minnet. Deretter fletter man sammen postene fra disse blokkene: Nye blokker fra hver subliste hentes inn ved behov, fulle (ferdigsorterte) blokker skyfles videre til den/de prosessene som bestilte sorteringen.

Kostnad: Hver blokk leses til minnet og skrives tilbake til disk i fase 1, dvs. 2 I/O pr. blokk.

I fase 2 leses hver blokk til minnet én gang. Hva de ferdigsorterte blokkene deretter skal brukes til, varierer med situasjonen: Kanskje er dette (TPMMS) del av en større prosess, i såfall går disse blokkene videre til en ny algoritme (pipelining el.l.), eller de skal simpelthen lagres på disk. Vanskelig regner vi ikke denne siste I/O-en som en del av algoritmens kostnad.

Totalt: 3 I/O pr. blokk.

b) Det som avgjør bruk av TPMMS kontra andre sorteringsalgoritmer, er størrelsen på det som skal sorteres:

Hvis hele relasjonen kan rommes i minnet samtidig, brukes quicksort eller noe annet tilsvarende.

Hvis ikke, brukes TPMMS eller en annen algoritme som er beregnet brukt i denne situasjonen. Hvis antall dellister under TPMMS er så høyt at ikke alle listene kan representeres ved én blokk i minnet samtidig, må det flere faser til – da brukes trefase MMS etc.

Ekstraoppgaver

A. La oss si at vi slår sammen Hendelse og Epikrise i én tabell.

(i) Hvordan vil tabellen se ut?

(ii) Hvilke FDer gjelder?

(iii) Hvilke kandidatnøkler har den?

(iv) Hvilken normalform er den på?

(i) HE(hendID, salgsID, dato, status, info)

(ii) hendID → salgsID (fra primærnøkkelen i Hendelse)

hendID, dato → status, info (" i Epikrise)

(iii) hendID og dato må være med i alle kandidatnøkler fordi de ikke forekommer i noen høyreside. Siden

$(hendID, dato)^+ = hendID, dato, salgsID, status, info$

er (hendID, dato) kandidatnøkkel (den eneste slike).

(iv) hendID → salgsID : hendID er ikke supernøkkel, så bryter BCNF.
salgsID er ikke nøkkelattributt, så bryter 3NF.
hendID er ekte delmengde av kandidatnøkkelen, så bryter 2NF.

Totalt på 1NF, bryter 2NF.

Behøver da strengt tatt ikke undere den siste FDen, men gjør det likevel:

hendID, dato → status : Venstresiden er supernøkkel, så er på BCNF.

hendID, dato → info : Samme her.

Totalt blir HE på 1NF, men bryter 2NF.

Ekstraoppgaver

B. Vi skal se på en liten utvidelse av akvariebutikkdatabasen. Butikken har flere rabattordninger - bl.a. en for de som har handlet for mer enn en viss sum foregående år, og en for de som er medlem i Pirajafiskens venner. En kunde kan være med i flere rabattordninger, men kan bare bruke én rabattordning i forbindelse med hvert kjøp.

Til å håndtere dette, har databasen tabellen

Rabatt(kundeID, rabattnavn, salgsID, prosent)

der rabattnavn er navnet på en rabatt kunden har, salgsID er et salg der kunden har benyttet denne rabatten og prosent er hvor stor rabatt denne kunden har for akkurat denne rabattypen.

(i) Hvilke FDer gjelder?

(ii) Hvilke kandidatnøkler har Rabatt?

(iii) Dekomponer Rabatt til BCNF

Ekstraoppgaver

Løsningsforslag B.

(i) Fra før har vi at

salgsID → kundeID

Dette bør jo gjelde i den nye tabellen også.

Siden det er maksimalt én rabattordning som kan brukes pr. kjøp, har vi dessuten at

salgsID → rabattnavn

(Men vi har ikke kundeID → rabattnavn, for en kunde skulle kunne ha flere rabatter.)

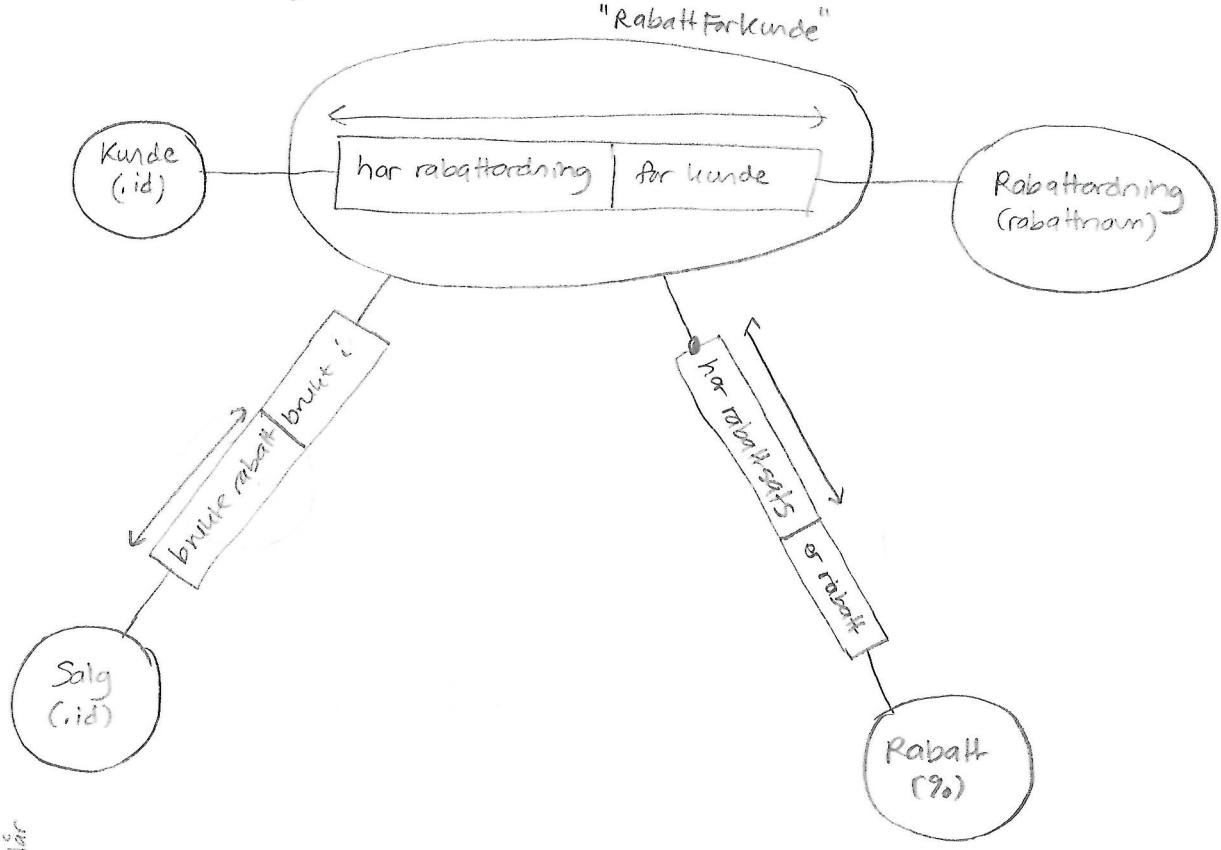
Dessuten avhenger størrelsen på rabatten på hvilket rabattprogram og hvilken kunde det er snakk om, så

kundeID, rabattnavn → prosent

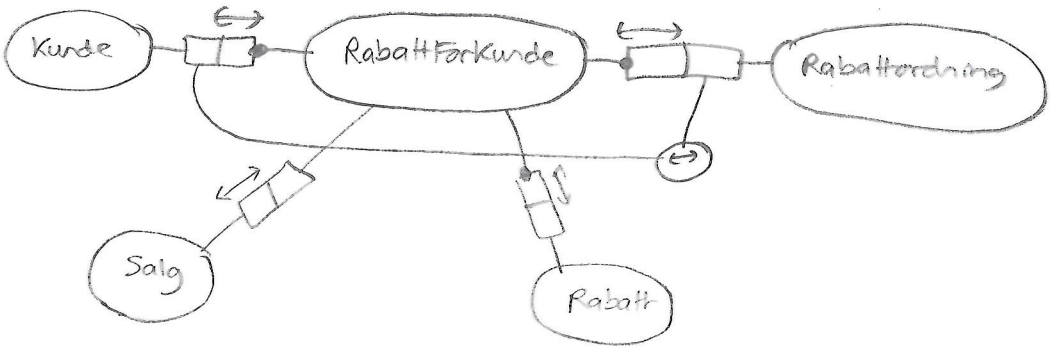
(Men ikke rabattnavn → prosent, for da kan man ikke ha forskjellige prosenter for forskjellige kunder innen en rabatttype.)

Til B (i)

De som har Inf130, vil kunne ha nytte av å modellere og gruppere, da vil FDere fremkomme nærmest av seg selv hvis modellesingen er riktig:



eller, skrevet fullt ut (droppet rollenavn og referansenåter):



Gruppert:

RabattForkunde(<u>kundeID</u>, <u>rabattnavn</u>, prosent)

RabattSalg(<u>salgsID</u>, kundeID, rabattnavn)Ⓐ

Dermed:

kundeID, rabattnavn → prosent

salgsID → kundeID, rabattnavn

Ⓐ Egentlig skulle RabattSalg vært slått sammen med den gamle Salg, og vi skulle i tillegg ha angitt eksklusivitet stier mellom salg – kunde og salg – RabattForKunde-kunde slik at de to kundeID'ene fra gammel og ny versjon blir slått sammen til én under gruppering, men for å tydeliggjøre denne delen av oppgaven, innfører jeg i stedet en ny relasjon RabattSalg. (Og understår eksklusivitet stier.)

B (forts.)

(ii) salgsID må være med i alle kandidatnøkler fordi den ikke er i noen høyreside.

$salgsID^+$ = salgsID, kundeID, rabattnavn, prosent

Så (salgsID) er kandidatnøkkel (eneste slike).

(iii) Må først vurdere hvilke brudd vi har:

salgsID → kundeID — er på BCNF; venstresiden er kandidatnøkkel og derfor supernøkkel

salgsID → rabattnavn — ⊣

kundeID, rabattnavn → prosent — er på 2NF, men bryter 3NF:
venstresiden er ikke supernøkkel,
høyresiden er ikke nøkkelattributt,
venstresiden er ikke en ekte delmengde av en kandidatnøkkel.

Rabatt(kundeID, rabattnavn, salgsID, prosent)

splitter etter kundeID, rabattnavn → prosent

RabattForKunde(kundeID, rabattnavn, prosent)

FD'er i denne:

kundeID, rabattnavn → prosent

Er på BCNF, lokal kandidatnøkkel er (kundeID, rabattnavn)

RabattSalg(kundeId, rabattnavn, salgsId)

FD'er i denne:

salgsId → kundeId, rabattnavn

Er på BCNF, lokal kandidatnøkkel er (salgsID).