Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

LØSNINGSFORSLAG TIL

EKSAMENSOPPGAVE I FAG TDT4145 – DATAMODELLERING OG DATABASESYSTEMER

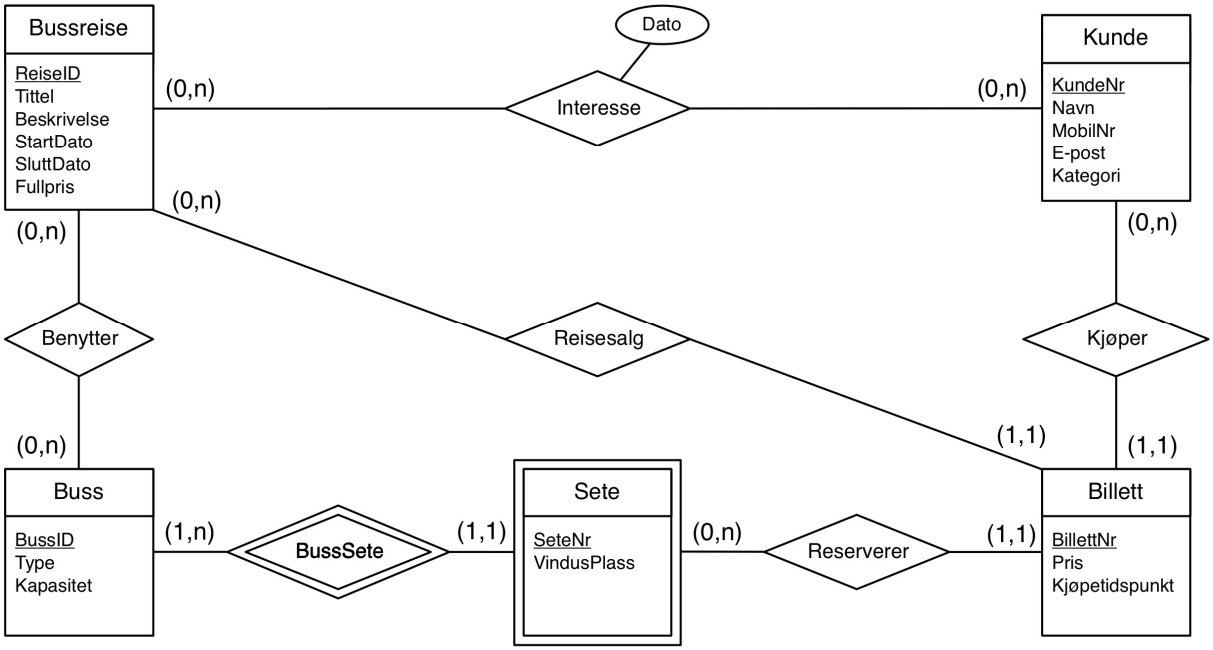
Eksamensdato: **4. juni 2012**

Eksamenstid: **15.00-19.00**

Tillatte hjelpemiddel: **D: Ingen trykte eller håndskrevne hjelpemiddel tillatt. Bestemt, enkel kalkulator tillatt.**

Oppgave 1 – Datamodellering – 22 %

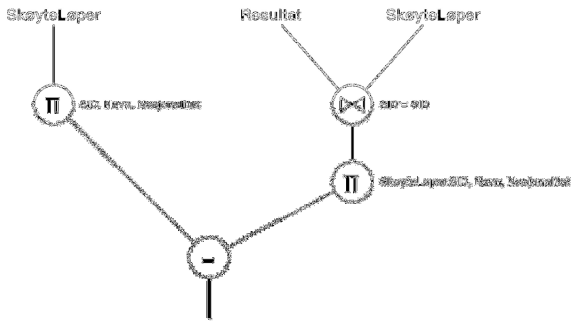
ER-diagram med nøkkelattributter og attributter på relasjonsklassene:



I forhold til den tekstlige beskrivelsen har vi her lagt til attributtene VindusPlass og Kjøpetidspunkt, det er ikke nødvendig å gjøre dette for å få full uttelling. Det kan finnes andre løsninger som er like gode eller bedre.

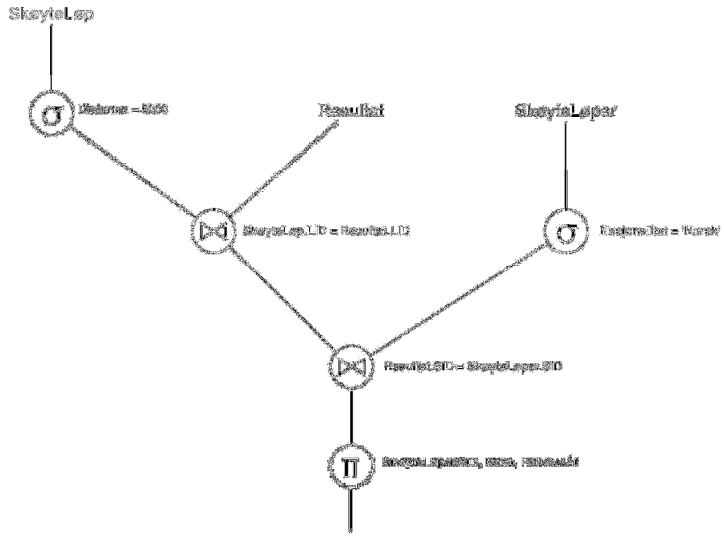
Oppgave 2 – Relasjonsalgebra og SQL – 20 %

a) Relasjonsalgebra:



Oppgaven kan løses på andre måter som er like gode, for eksempel ved å bruke anti-join-operatoren.

b) Relasjonsalgebra:



Oppgaven kan løses på andre måter som er like gode, for eksempel ved å bruke naturlig-join-operatoren.

c) SQL:

```
select DISTINCT SL.SID, Navn, Nasjonalitet
from SkøyteLøper as SL, Resultat as R, SkøyteLøp as L
where SL.SID = R.SID
  and R.LID = L.LID
  and L.Distanse = 5000
  and not exists (select *
                  from Passeringer as P
                  where P.LID = L.LID
                    and P.SID = SL.SID
                    and P.RundeTid >= 30)
```

Oppgaven kan løses på andre måter som er like gode, for eksempel kan foreningen gjøres i from-delen av setningen.

d) SQL:

```
select LID, count(*)
from Resultat
group by LID
having count(*) > 20
order by count(*) desc
```

Oppgave 3 – Teori – 15 %

a) Tillukningen, X^+, til en mengde attributter, X, er mengden attributter som er funksjonelt avhengig av attributtene i X. ab^+ = abcde.

b) En forening av R_1 og R_2 vil gjøre at alle b-verdier som er assosiert med en a-verdi, blir kombinert med alle (cdef)-verdier som er assosiert med samme a-verdi. For at dette skal være tapsløst må vi ha multi-verdi avhengighetene (mvd): a ->> b og a ->> cdef.

c) Referanseintegritet vil si at alle fremmednøkler enten referer til et tuppel som finnes eller har null-verdi (ikke referer til noe).

Oppgave 4 – Mapping – 5 %

Et relasjonsskjema vil ha en tabell for entitetsklassen A med alle attributter, og en tabell for entitetsklassen B med alle attributter.

Relasjonsklassen R kan legges til i tabellen for entitetsklassen på mange-siden av relasjonen, altså som ekstra attributter i tabellen for A og med en fremmednøkkel mot B. En alternativ løsning er å lage en egen tabell for R, med attributtene til relasjonsklassen og fremmednøkler mot A- og B-tabellene.

Dersom $\alpha = 1$ vil alle A-entiteter delta i en R-relasjon. Det taler for den første løsningen. Når $\alpha = 0$ trenger ikke A-entiteter å ha en R-relasjon. I slike tilfeller kan det være aktuelt å velge den andre løsningen. Tre forhold som kan diskuteres er mengden null-verdier, behovet for å forene (joine) tabeller og den fysiske størrelsen på A-tabellen.

Oppgave 5 -- Lagring, indekser og queryutføring – 25 %

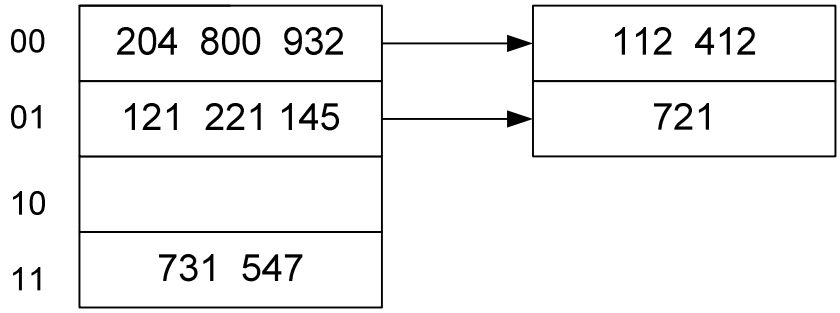
a) **Statisk hashing**.

Vi lager 3 bits hashverdier som vi bruker i a) og b).

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

I oppgave a) bruker vi bare de to siste bitene (MOD 4):

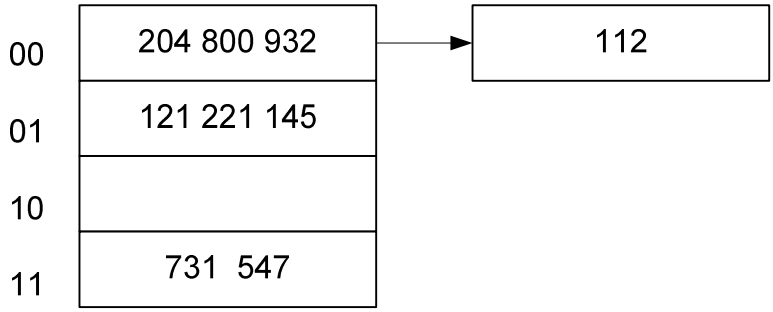


Gjennomsnittlig antall blokker er (8*1 + 3*2) / 11 = 1.27

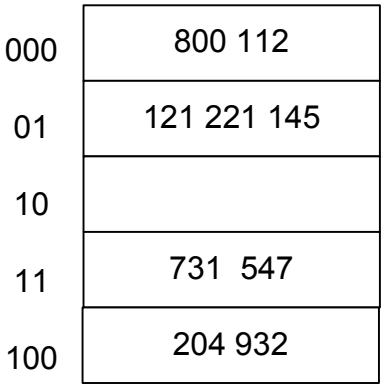
I læreboka har de brukt delte overløpsblokker med overløpspekere direkte til postene. Dette er også en OK løsning til dette spørsmålet som gir samme gjennomsnittlig antall blokker aksessert.

b) **Lineær hashing**:

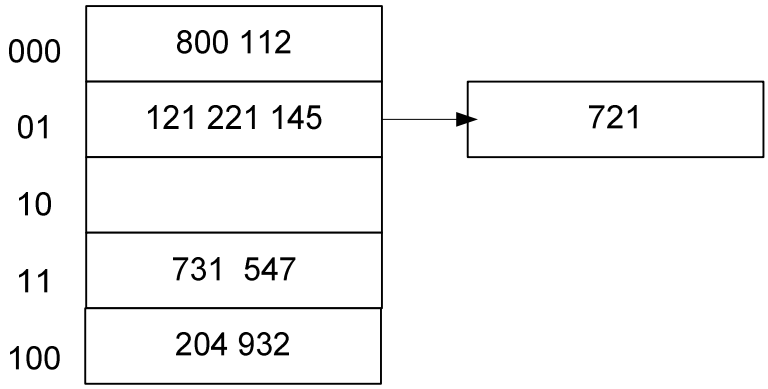
Her starter vi med 4 blokker og MOD 4. I oppgaven var det ved en feil opplyst MOD 8, men dette ble det informert om på eksamen like etter start av eksamen. Vi fyller på med poster inntil den første overflytsblokka er laget.



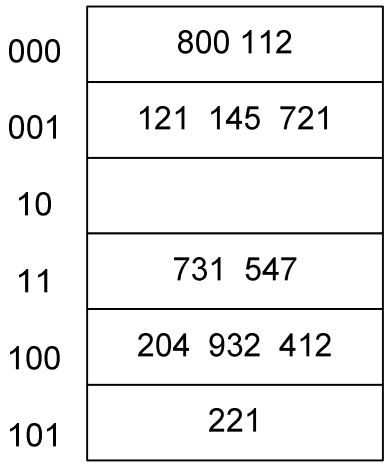
Nå utvider vi hashfila med ei blokk og må ta i bruk et bit mer av hashfunksjonen for n<1. Da flytter noen poster (204 og 932) seg fra 00 til 100.



Neste overflytsblokk er:



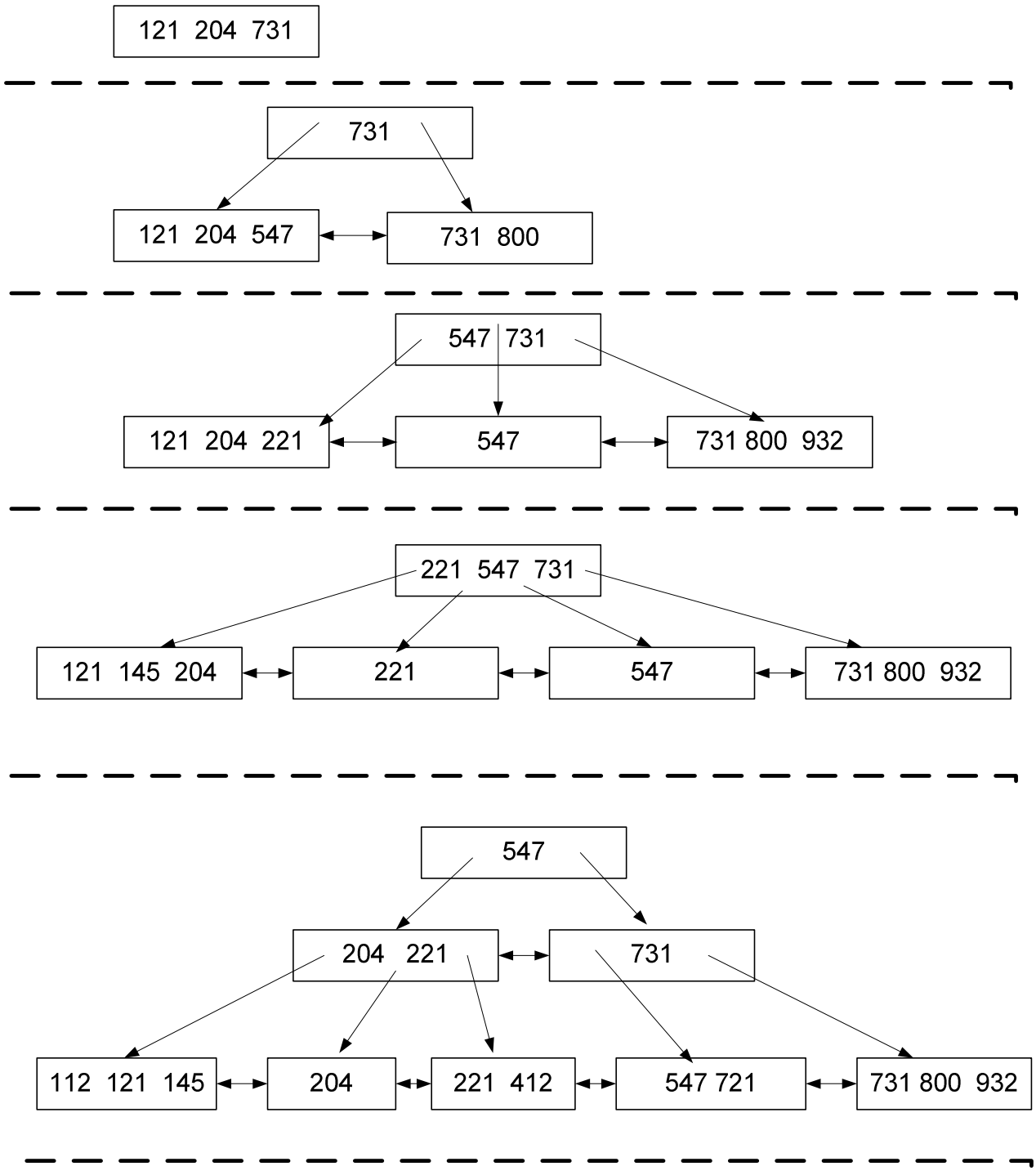
Så blir sluttilstanden (**det er spurt om**):

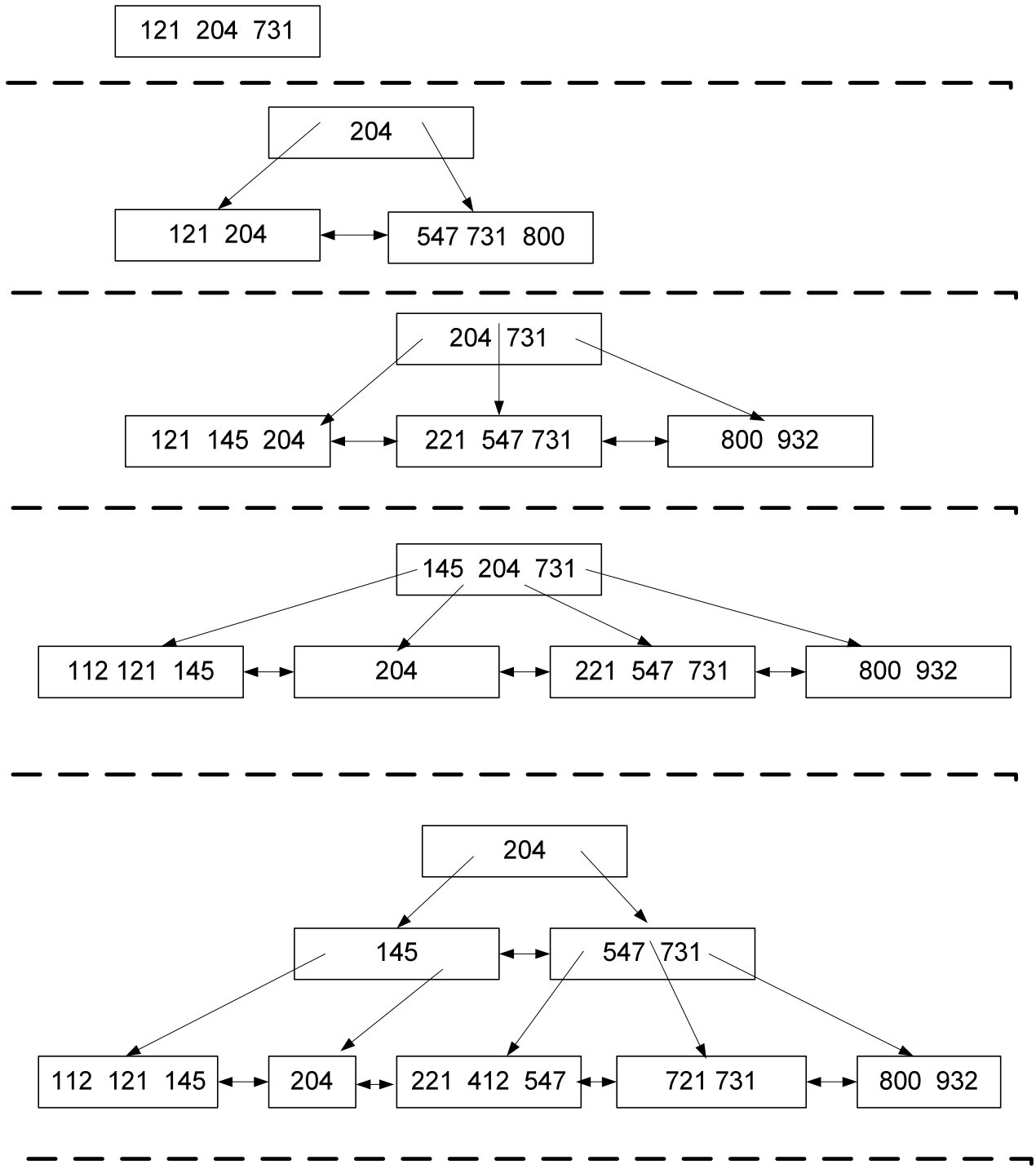


c) **B+-trær**:

Her har vi vist to alternative versjoner av splittingen. Den første viser den klassiske versjonen av B+-trær, hvor minste element i høyreblokka blir splitteelement, mens versjon to viser Elmasri/Navathe-versjonen hvor det største elementet i den venstre blokka blir splitteelement.

Ved en splitt lager vi først plass ved å splitte den eksisterende blokka uten å se på den nye nøkkelen, etterpå setter vi inn den nye nøkkelen der den hører hjemme. Generelt er det lurt å lage all plass man trenger før man forsøker å utføre selve operasjonen.


121 204 731
731
121 204 547
731 800
547 731
121 204 221
547
731 800 932
221 547 731
121 145 204
221
547
731 800 932
547
204 221
731
112 121 145
204
221 412
547 721
731 800 932



d) **Join**: Vi bruker 1 bufferplass til resultat. Vi må dele Department i to deler og bruker 6 plasser til Department og har da 3 plasser til Employee. Vi får: 12 (hele Department)+ 2*2000 (hele Employee) = **4012** blokker som må leses.

Oppgave 6 – Transaksjoner – 20 %

a) **Rollback:**
BeforeImage blir satt inn igjen for B og D, slik at vi får A: 30, B: 20, C: 40, D: 25.

b) **REDO:**

102: Trenger ikke REDO da recLSN = 104. Vi starter redoscannet på 104, da det er eldste loggpost som kanskje trenger REDO utifra tilstanden til DPT (eldste recLSN i DPT).

104: Trenger kanskje REDO da recLSN = 104, leser inn blokka og ser at pageLSN = 104. Trenger ikke REDO.

106: Trenger ikke REDO da recLSN = 107.

107: Trenger kanskje REDO da recLSN = 107. Leser blokka og ser at pageLSN = 106. Gjør da REDO: Skriver inn AfterImage (D=26) og oppdaterer pageLSN til 107.