TDT4145 Øving 4

Kristoffer Dalby og Thor Martin Abrahamsen

Oppgave 1

a

Supernøkkel er en samling av attributter hvor kombinasjonen av attributtene alltid vil være unik.
Nøkkel er en minimal supernøkkel, det vil si at du ikke kan fjerne en attributt og fortsatt ha en supernøkkel som er en unik kombinasjon.
Funksjonell avhengighet er en begrensning mellom to sett av attributter fra databasen. Det betyr at en attributt eller et sett med attributter bestemmer en annen attributt eller et annet sett med attributter

b

Tillukningen av X er alle mulige funksjonelle avhengigheter som kan lages ut fra de funksjonelle avhengighetene som er gitt.

Algoritme:
Steg en, lag en liste med alle funksjonelle avhengigheter som er gitt. Steg to, for alle avhengighetene gitt, hvis venstre side av avhengigheten er i listen og høyre siden ikke er, legg høyre side i listen. Steg tre, Fortsett til listen ikke endress.

c

$a+ = a, e$
$ab+ = a, b, e, c, d$
$e+ = e$

d

Man finner en supernøkkel ved å analysere de funksjonelle avhengighetene. Dette er i allhovedsak beregning av tillukningene til kandidat settet for det vi lurer på om er en supernøkkel. En supernøkkel er en nøkkel om den er minimal, det vil si at det ikke lenger kan fjernes attributter uten at den har supernøkkel status. Med andre ord, en nøkkel er en supernøkkel som er minimal.

e

Hvis de like attributtene i begge de nye projeksjonene er supernøkkel utgjør en supernøkkel for begge projeksjonene vil har begge tabllene en tapsløs-join egenskap.

f

$R1 \cap R2 = (bc) \neq R1 eller R2$ og er derfor ikke tapsløs.
$R1 \cap R2 = (bcd) = R2$ og er derfor tapsløs.
$R1 \cap R2 = (bc) = R2$ og er derfor tapsløs.

g

Et relasjonskjema R er 3NF hvis alle ikke primærnøkkel attributter av R møter begge disse kravene:

- Den er ikke en full funksjonell avhengighet av alle nøkler i R.
- Den er ikke transetiv for alle avhengigheter av alle nøkler i R.

h

Dette er en god dekomponering, da den er tapsløs. Siden vi kan se at R3 er en relasjon mellom R1 og R2 og vi kan dermed finne alle attributtene.

Oppgave 2

a

Oppgaven oppgir at en student har en 4 byte nøkkel og en 4 byte blokkidentifikator som resulterer i 8 byte per student. En blokk har 4096 byte og vi bruker dette og den gjennomsnittlige fyllgraden av en blokk på 0.67 til å regne ut antall studenter per blokk.
$\frac{4096}{8} x 0.67 \approx 343$ som er antall studenter per blokk. Vi tar derretter å finner ut det totale antall blokker:
$\frac{200000}{343} \approx 583$
Da blir det til slutt slik at det øverste nivået vil ha en blokk, som peker på to navigasjonsblokker under. Disse to blokkene vil peke på de resterende 583 blokkene. Dette fordi det kun er plass til 343 i en blokk.

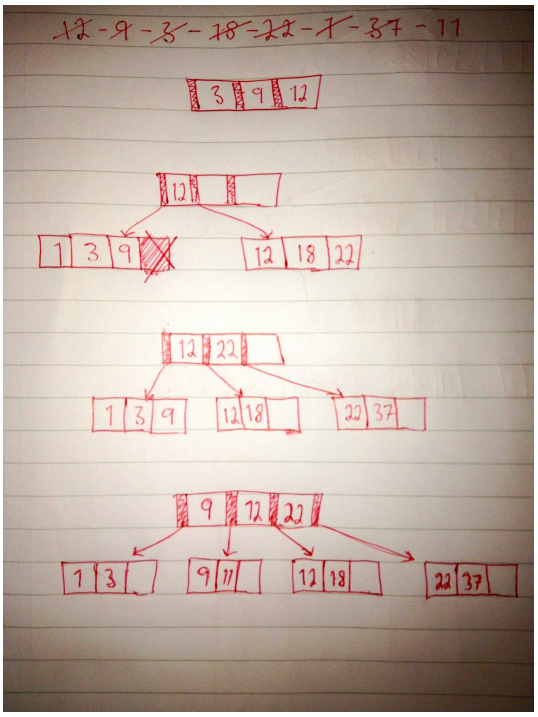
b

Det blir 6 diskaksesser, først vil vi lese rotnoden, etterfulgt av nivå en, søke gjennom blokkene, også skrive til løvnode og til slutt oppdatere nivå en noden og rotnoden. Blokkene vil ikke ligge bufferet i minne, men må leses fra disk.

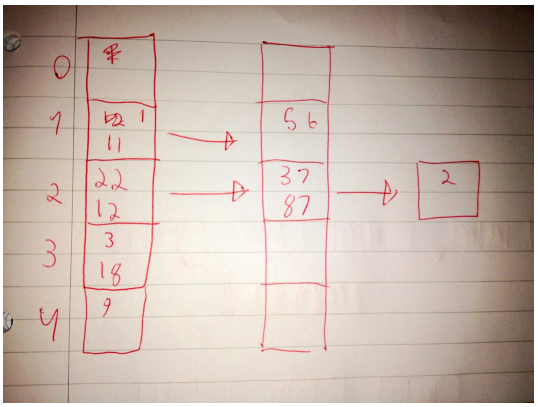
c

Ettersom vi kun skal oppdatere en node trenger vi ikke å oppdatere rotnoden og noden på nivå en. Dette gjør at vi totalt får fire diskaksesser. Vi antar også at systemet vår har harddisker med 4,5 ms søketid og solid state drives med 0,1 ms søketid.
Søketiden blir derfor $4.5 \times 4 \equiv 18ms$ for harddisken og $0.1 \times 4 \equiv 0.4ms$ for SSDen.

d


12 - 9 - 3 - 18 - 22 - 7 - 37 - 11
3 9 12
12
1 3 9
12 18 22
12 22
1 3 9
12 18
22 37
9 12 22
1 3
9 11
12 18
22 37

Oppgave 3


0
1
2
3
4
11
22
12
3
18
9
56
37
87
2

Oppgave 4

a

Vi tolker utifra oppgaven at de spørringene i hovedsak vil vektlegge at vi kan bruke vinnavn som en nøkkel, vi ser derfor at hashing vil være å foretrekke ettersom oppslag i en hashtabell er svært raskt. Det vil i tilfelle også ikke være noe verre tilfelle en at man må søke sekvensielt hvis det er flere tabeller til nøkkelen.

b

I oppgaven er det ikke oppgitt noen operasjoner på avdelingsrelasjonen. Vi antar også at det ikke derfor ikke gjøres noen store operasjoner på avdelinger, kanskje annet en en liten oppdatering i ny og ned og noen seleksjoner.

Vi ser at de mest vanlige spørringene er på ansatt-tabellen, med seleksjoner og innsettinger. Det gjøres sammenligninger på attributtene personr", navn"og "avdnavnså det vil være hensiktsmessig å indeksere disse. (navn"og "avdnavnéttersom Q2 blir mest brukt). Vi ville benyttet en hash-indeks, optimalisert for direkteaksess og ekvivalensseleksjon.