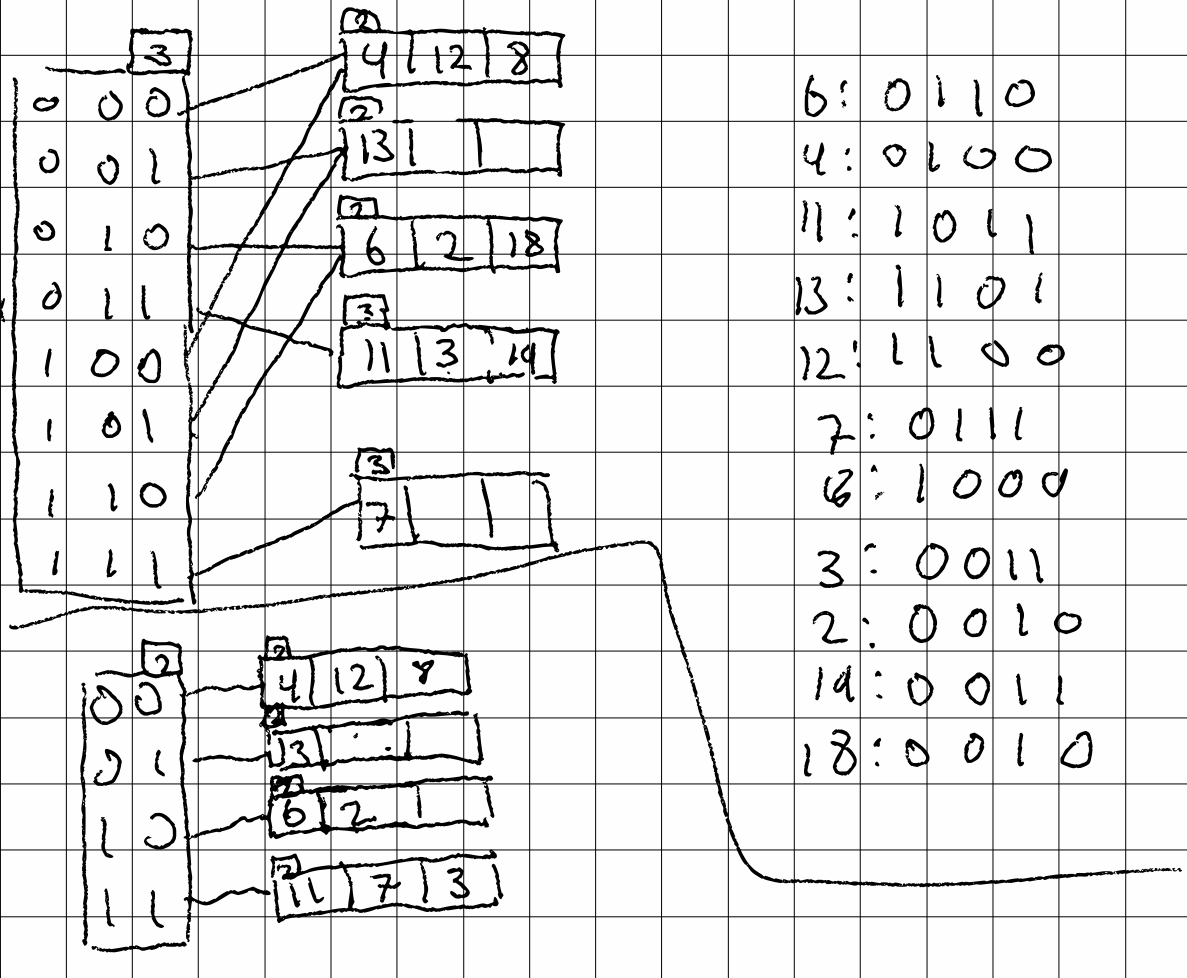
1) Extendible hashing

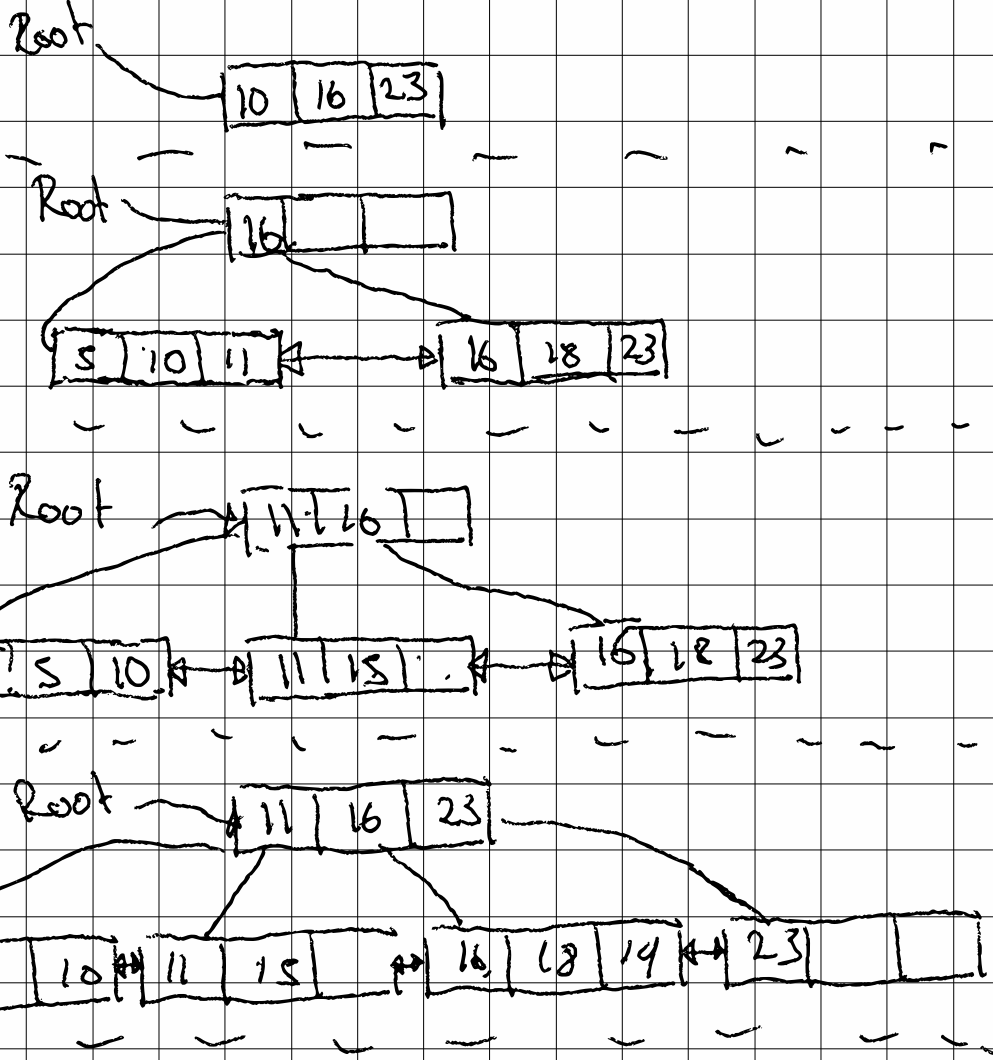


6: 0 1 1 0
4: 0 1 0 0
11: 1 0 1 1
13: 1 1 0 1
12: 1 1 0 0
7: 0 1 1 1
8: 1 0 0 0
3: 0 0 1 1
2: 0 0 1 0
19: 0 0 1 1
18: 0 0 1 0

For dir. double

Bom

2) Konstruksjon av B+ -tre




Root
19
11
16
23
3
5
10
11
15
16
18
19
20
23
25

3) Lagring og queries

10 000 pers, 1 post 250 byte per
Postene lagret i blocker på 2KB

8 poster per block

a) $\frac{10000}{8 \cdot \frac{2}{3}} = \underline{1875}$, 10000, og hver block fylles med 67% av 8.

b) Person ID 4 byte
Block ID 4 byte

L1: $\left\lfloor \frac{2048}{8 \cdot \frac{3}{2}} \right\rfloor = 170$, plass til 170 (PersonId, BlockId) i en block.

$\left\lceil \frac{1875}{170} \right\rceil = \underline{12}$, 12 blocker for å nå alle 1875 på lavnivå.

L2: $\left\lceil \frac{12}{170} \right\rceil = \underline{1}$ Kan nå alle 12 L1 blocker fra én på L2.

3) c)

1) 3 siden vi har sekvensregel.

2) 2 ned + 1875 bortover = 1877

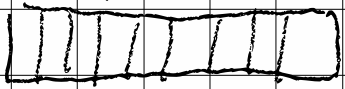
3) 2 ned + 1875 bort = 1877,
allerede sortert!

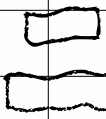
4) $\lceil 2 \text{ ned} + 1875 \cdot 0{,}05 \text{ bort} \rceil = 96$
kun 5% av postene tilfredsstiller betingelsen.

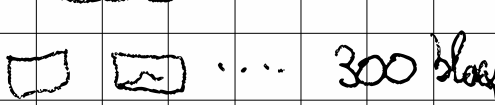
4)

(LastName, BlockId)

Heap 1250 blocker







1. 1250, fordi ikke sorteret i Bt pga. SELECT *

2. Ingen sorteringskrav: 1250 blocker
Scan heapfil.

3. 3 ned + 1 per "sorteret" i heap
Siden vi har sorteret i query.

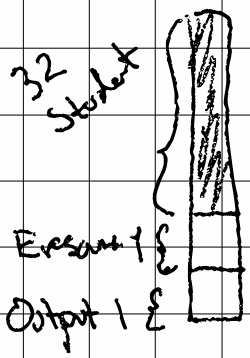
4. 2 ned + 300 bork = 302
Index-only query. Ser kun på Bt-tre.

5. 3 ned + 1 write Bt + 4 read heap + 1 w heap
4 for at finde blokke og sætte ind + 2 for at læse / skrive til heapfil. 6

5) Nested-loop join

S: 47 000 posters
i 800 blocks

E: 500 000 posters
i 12 800 blocks



$\frac{800}{32} \cdot (32 + 12800)$

$= \underline{320\ 800}$

6) Transaksjoner

a) Vi ønsker støtte for deling og samtidig adgang av data.

Vi ønsker sikker, pålitelig og atomisk adgang til store mengder data.

b) Atomic: enten kjører de fullstendig eller kjører de ikke

Consistency: overholder konsistenskrav (PK, refrences, check ...)

Isolation: isolert fra hverandre, merker ikke at hverandre kjører.

Durable: Er permanente. Mistes ikke etter commit.

c) Recoverable:
Her transaksjon committer etter transaksjoner de har lest fra har committet.

ACA:
Når transaksjoner kun kan lese verdier skrevet av committede transaksjoner

Strikt:
Når transaksjonene verken kan lese eller skrive inne-committede verdier

H_1: Recoverable

H_2: $w_3(x)$ o_3 $w_1(x)$ feil c_3 her committa.
ACA

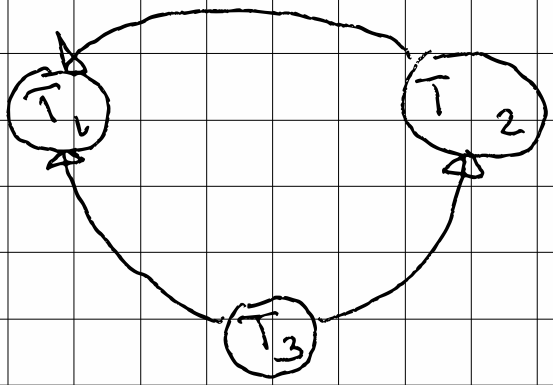
H3: Unrecoverable, c_2 eller c_3.

d) Hvis

1. De tilhører forskjellige transaksjoner.
2. De bruker samme dataelement.
3. Minst én av operasjonene er en write.

Altså hvis endring av rekkefølgen endrer resultatet på databasen.

e)



Ingen syklr => konfliktserialiserbar

f) Vi har en vranglås (deadlock) når to (eller flere) transaksjoner venter på hverandres låser.

g)

T_1	T_2	T_3
rl(Y)		
r(Y)		
		wl(x)
		w(x)
		rl(y)
		r(y)
	trylock(x)	
trylock(x)		
		commit /
		unlock(x,y)
	wl(x)	
	w(x)	
	commit/	
	unlock(x)	
rl(x)		
r(x)		
commit/		
unlock(x,y)		

7) a) Vinnie: T1, T2 committet

Tapse: T3, rullet ikke committe

b) Transaksjonstabell

T_n	Status	Last_LSN
T_1	committed	173
T_2	committed	170
T_3	in progress	174

DPT

Page Id	Rec_LSN
A	168
B	169