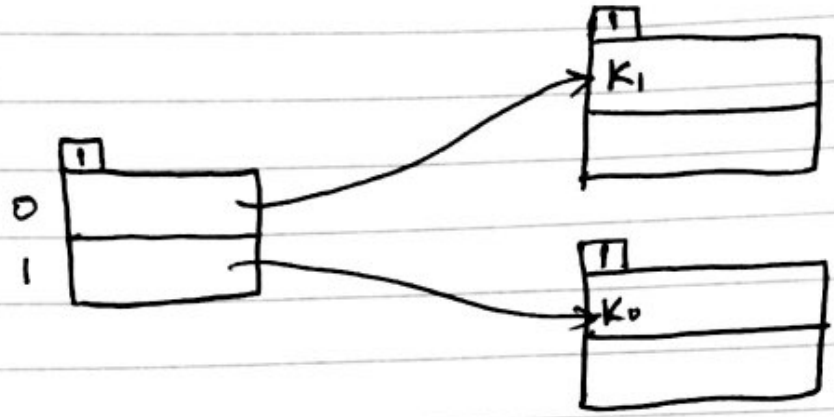
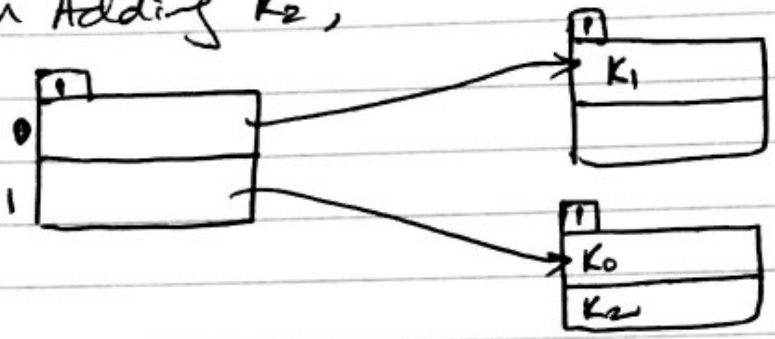
HOMEWORK 4

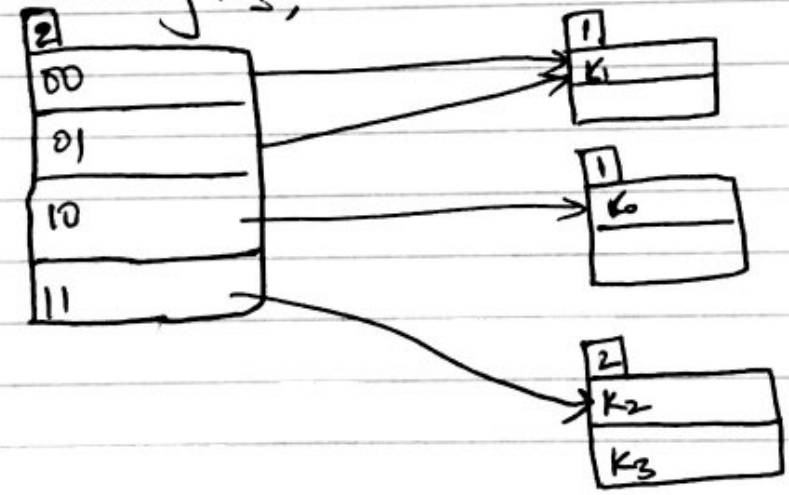
Q1.)



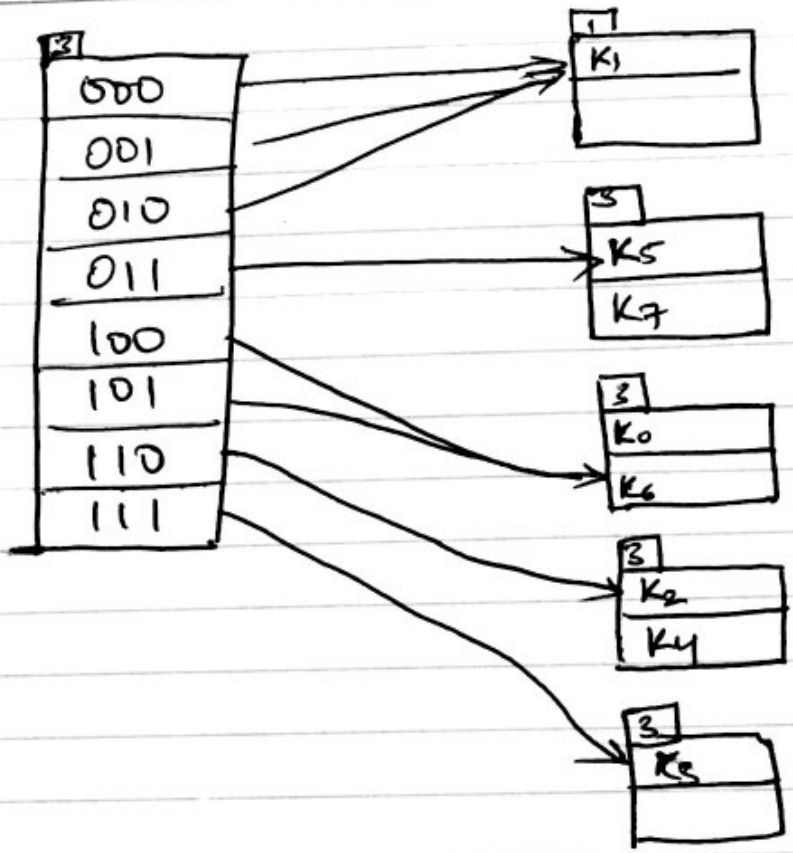
On Adding K_2,



On Adding K_3,



Similarly on adding K_4, we have to increase the keys further in the table.



Q2)

RPM = 150,00

Tracks = 250,000

2000 – Sectors

Sector Size = 512 bytes

Capacity of disk = $2 \times 2 \times 250,000 \times 2000 \times 512$

= 1.024 TB

Size of each track = 2000×512 = 1.024 mb

So, 1mb can be read in 1 rotation

∴ Maximum rate at which data can be read from disk

$= \frac{150000 \times 1.024}{60} = 256$ mb/s

Average rotational latency = $0.5 \times 1000 \times \frac{60}{15000}$

= 2ms

(b) Seek Time = 5ms
Block Size = 4kb

Transfer Rate = $\frac{4\text{ kb}}{250\text{ mb/s}}$ = 0.016 ms

Time to Read = Seek Time + Rotational latency + Transfer Rate

= 5 + 2 + 0.016
= 7.016 ms

BLOCK MODEL

For file size of 100 Kb,

Blocks = $\frac{100\text{Kb}}{4\text{ Kb}}$ = 25

∴ Transfer Rate = 7.016 × 25 = 175.4 ms

For file size of 1000 Kb,

Blocks = $\frac{1000\text{Kb}}{4\text{Kb}}$ = 250

∴ Transfer Rate = 7.016 × 250 = 1754 ms

For file size of 100 MB,

$$\text{Blocks} = \frac{100\ MB}{4\ KB} = 25{,}000$$

$\therefore$ Transfer Rate $= 7.016 \times 25000 = 175400$ ms

LATENCY / TRANSFER MODEL

For file of size 100 KB,

$$t_r = 2 + 5 + \frac{100}{256} = 7.3 \text{ ms}$$

For file of size 1000 KB,

$$t_r = 2 + 5 + \frac{1000}{256} = 10.9 \text{ ms}$$

For file of size 100 MB,

$$t_r = 2 + 5 + \frac{100{,}000}{256} = 397.625 \text{ ms}$$

(c)

File Size = 36 GB

Main Memory = 1 GB

We can do 36 files of size 1 GB each

$= 36 \times$ (Time to sort 1 file)

$$= 36 \times \left[5 + 2 + \frac{1 \times 10^3}{256} \right] * 2 = 28135$$

Considering 36 splits;

Buffers = 36 + 1 (o/p buffer)
= 37

Buffer Size = $\frac{1GB}{37}$ = 27.02 MB

Total files = $\frac{36}{27.02 MB}$ = 1333 files

Time to read & write these files = $1333 \times \left\lceil \frac{5+2+27.02}{256} \right\rceil 2$

= 306.8 s

Total Time = 306.8 + 281.75 s
= 588.55 s

(d) d=6 ⇒ 2 phases.

Buffer = 6(input) + 1 (o/p) = 7
Buffer size = $\frac{1GB}{7}$ = 142.86 MB

Total files. $\frac{36 GB}{142.86 MB}$ = 252

Time to read & write these files = $252 \left\lceil \frac{5+2+142.86}{256} \right\rceil 2$

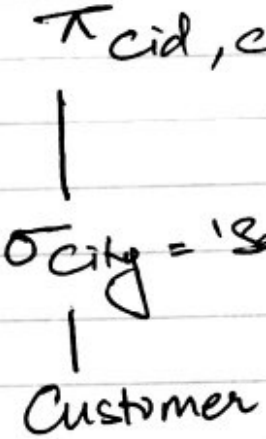
= 284.78 s

Total Time = $284.78 \times 2 = 569.56$ s

Q3)

(a) Query 1

Query Tree —

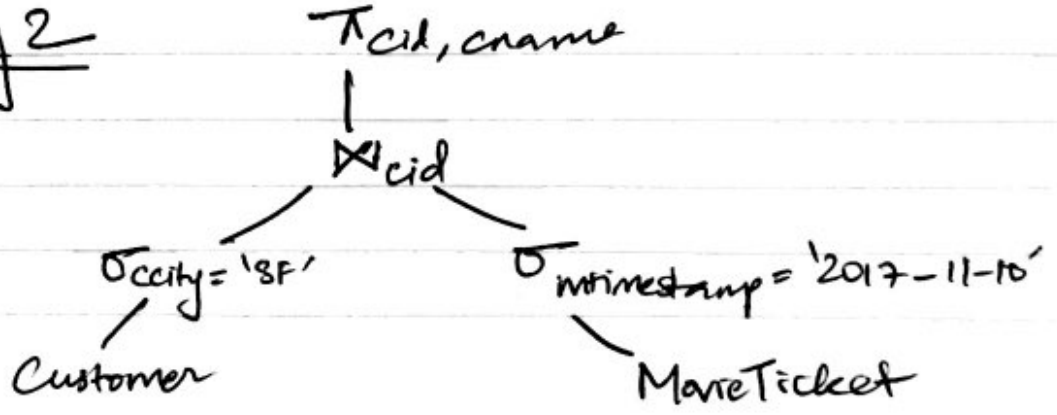


Total number of customers in SanFrancisco = 1 million
Main Memory size = 1.5 GB
Main memory data = 1 million x 100 bytes (tuple size) = 100 mb

Now, because there is no index we dont know when to stop looking for customers = 100 mb x 100 byte = 10GB

Time to read = 10ms + $\frac{10\,Gb}{150\,mb/s}$ = 66.67 s

Query 2



Here we join 2 tables customer and movieticket where city is San francisco and date 2017-11-10.

To scan for SF take 66.67s
Movie Ticket has 5 billion rows of 40 bytes.

Cost = 10ms + $\frac{5 \text{ billion} \times 40 \text{ bytes}}{150 \text{ mb/s}}$ = 1333333s

Total Time = 66.67 + 1333333 = 1400s

Query 3

$\pi_{mid, mname}$

$\bowtie_{mid}$

$\sigma_{type = \text{'Horror'}}$ (Movie)

$\sigma_{mtimestamp = \text{'2017-10-11'}}$ (Movie Ticket)

Cost = $\left(10ms + \frac{0.5 \text{ million} \times 100 \text{ bytes}}{150 \text{ Mbps}}\right) + \left(10ms + \frac{5 \text{ billi} \times 40 \text{ b}}{150 \text{ mb}}\right.$

= 0.34s + 1333.33s

= 1333.67s

(b) city size = 32 bytes
cid size = 8 bytes
RID = 8 bytes

Each Tree node has n-1 keys & n-pointers.

Since Customer table has a sparse index,

$(n-1) * 32 + n * 8 = 4096$

$n = 103$

Assuming occupancy of 80%, we will have 82 indexes.

Considering Customer table = 100 million
Entry size Customer = 100 bytes.
Records per disk page = 4096/100 = 40 records
No of indexes = $\frac{100 \text{ million}}{40}$ = 2.5 million
We will have 2.5 million blocks of 40 records.

No of leaf nodes = $\frac{2500,000}{82}$ = 30488 nodes

No of nodes in next level = $\frac{30488}{82}$ = 372 nodes

No of nodes in next level = $\frac{372}{82}$ = 4 nodes

So, the B+tree will have 4 level of nodes: the root, two internal levels and leaf level.

Size of tree = 30480 × 4KB = 121 MB

Time to fetch a single record = 5 × 10 ms = 50 ms

Time to fetch 50 records = (4+3) × 10 = 70 ms

DENSE UNCLUSTURED INDEX ON (CID):

Each Index Entry = 8 (cid) + 8 (RID)
= 16 bytes.

Nodes = $\frac{4096}{16}$ = 256

Assuming 80% occupancy, we will have 204 entries per node.

MovieTicket has 5billion records.

No of leaf nods = $\frac{5 \text{ billion}}{204}$ = 24.5 million

No of nodes in next level = $\frac{24.5}{204}$ = 120,000

No of nodes in next level = $\frac{120000}{204}$ = 588

No of nodes in next level = $\frac{588}{204}$ = 2 nods

So, the B+ tree will have 5 levels.

Size of tree = 24.5 million x 4KB = 98 GB

Time to fetch a single record = 6 x 10ms
= 60ms

Time to fetch 50 records = (49 x 10ms)
+ 60ms
= 550ms

(c)

Query 1 – I would choose a clustered index on cid and ccity in customer, to accelerate the process of fetching records from customers.
To fetch 1% of customer from SF it would take 01 sec.

Query 2 – Choosing clustered indexes on ccity in Customer and in timestamp in Movie Ticket would be ideal. We would first fetch users in SF which would take only 1% of the cost and then further filter for the date. Therefore the cost would be reduced further.

Query3 - We choose unclustured indexes on Movie in Movie and clustured on Movie Ticket. The first index can fetch users from SF, which would take only 1% of the cost. Then we can scan Movie Ticket for timestamp '2017-10-11', hence improving runtime.