NAME: NAMAN GOYAL

ROLL NO: 180123029

Sol^n 1) no. Surfaces = 10

no. of tracks = 10,000

⇒ Total no of tracks : $10000 \times 10 = 10^5$.

no of sectors = 1000, bytes per sector = 512

⇒ capacity of disk = 512 (Bytes) × 1000 × (Total no of tracks)

$= 512 \times 10^8$ Bytes $= \frac{512 \times 10^8}{2^{30}}$ GB

$\therefore$ 1 GB = 2^{30} Bytes.

$= 47.683$ GB

So, the capacity of disk = 47.638 GB.

Name: Naman Goyal
Roll_No. : 180123029

Camlin Page
Date / /

Soln 2: Using two-phase multiway merge sort,

Tuples to be sorted = 10^7

Capacity of each block = 16384 → No of sector/block = $\frac{16384}{512}$ = 32.

Time req^r/ rotat^n = 10^4 rev/mi = 6ms.

Transfer time of each block ⟹

Head must cross 32 sectors and 31 gaps as gap = 20/.. Degree of gap = $\frac{20 \times 360}{100} = 72°$

" of sector = $\frac{80 \times 360}{100} = 288°$.

⟹ Total degrees travelled by arc = $72 \times \frac{31}{1000} + 288 \times \frac{32}{1000}$

= 11.448°.

⟹ Transfer time = $\frac{11.448}{360} \times (0.006) = 0.0001908$ seconds.
= 0.0190 ms.

⟹ Avg. rotational latency = $\frac{1}{2}$ (Time of rotat^n) = 3ms.

∴ Avg seek time = Time req^r to move total track

$= 1 + (0.001)\left(\frac{10000}{3}\right)^3 = 4.34$ sec.

⟹ ∴ Avg. latency = Avg. seek time + Avg Rot. Latency + Transfer time

= (4.34 + 3 + 0.19) ms = 7.53 ms.

Camlin Page
Date / /

while applying two phase multi way merge sort:-

During Phase I: ∴ 6400 block will fill main memory.
we fill MM 16 times.
we read: 10^5 blocks once & write 10^5 new blocks once.
So there are 2×10^5 disk I/O's.

Thus, time taken (by Phase I) = $2 \times 10^5 \times$ (Avg. Latency)
$= 2 \times 10^5 \times 7.53 \times 10^{-3}$ sec.
$= 1506$ sec.

During Phase II: They perform same 2×10^5 disk I/O's.

Total time = 1506 sec + 1506 sec
$= 3012$ sec. $= 50.2$ mins.

So, total time = 3012 sec = 50.2 mins.

Name : Naman Goyal
Roll_No : 180123029

Camlin Page
Date / /

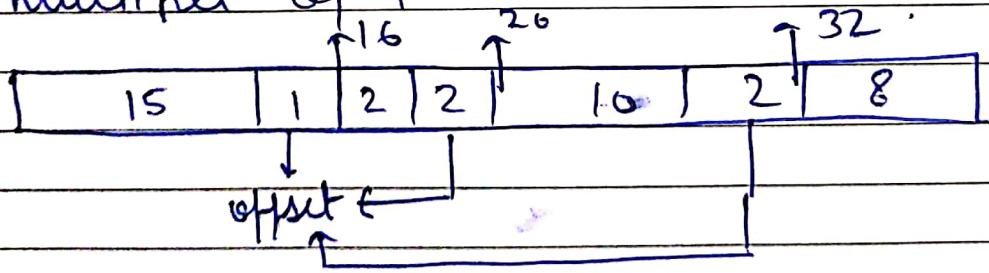
Sol^{n} (3):
The sizes of SQL (date and time) are 10 bytes and 8 bytes.

(1) As the fields can start at any byte, no. offsets are req^{r}.

15	2	10	8

Total no. of bytes req^{n} is 15+2+10+8 = 35 bytes.
⇒ The record takes 35 bytes.

(2) As the fields starts at multiples of 4: the offset at the end of each field are such that the next field starts at a byte multiples of 4.



Total no. of bytes req^{r} = 16 + 4 + 12 + 8 = 40.
⇒ The record takes 40 bytes.

Name: Naman Goyal
Roll No: 18012302 9

Camlin Page
Date / /

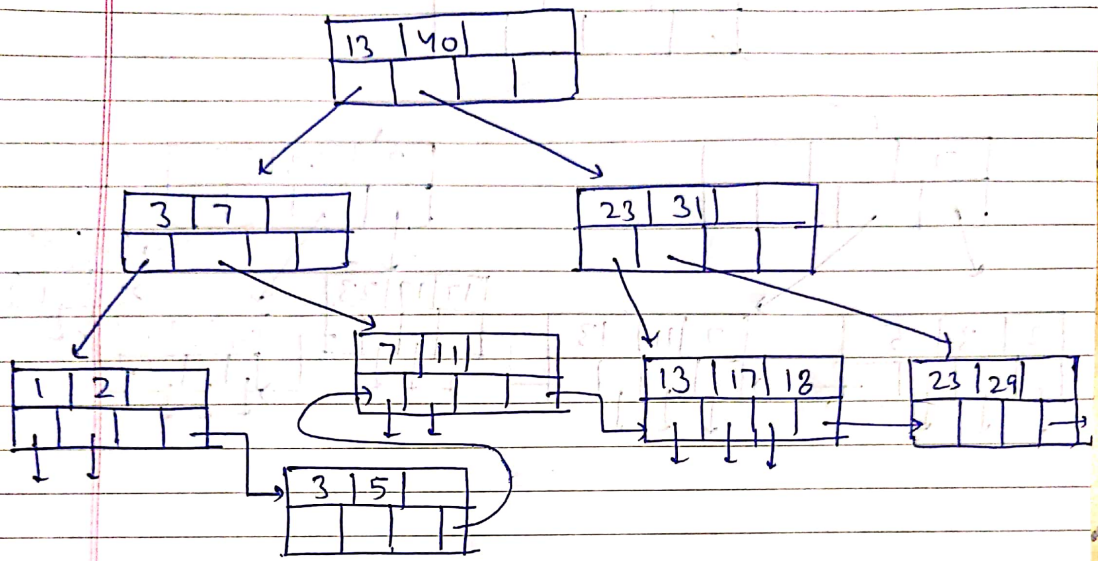
Soln ④ (a) Upon deletion of records with keys 60, 70, 80:
the first index block will have keys 10, 20, 50 with an empty slot.
The data record with 50, 60 will have only 50.
The second data block will have (20,21) along with
(22,23), (24,25), (26,27) (28,29) & (40,–) overflow blocks connected in sequence with second data block.

(b) Inserting Key 1 in B-Tree:

By lookup procedure, we find that insertion will go to first leaf of 3rd layer but since n=3, leaf can't have 4-key pointer pairs 1, 2, 3, 5. First we create a new node & move the highest two keys 3, 5 along with their pointers to that node. We insert a pointer to node (3,5). Then we must insert a key 3 in leaf with key 7 in 2nd layer and then insert a pointer in this leaf which points to leaf with keys (1,2) in 3rd layer and also insert a pointer which points to leaf (3,5) in the 3rd layer and connect all leaf by last pointer to its right.

graphical representation :–

Name: Naman Goyal
Roll No: 180123029



Sol^{n} (4) (b) (2) Upon deleting '23' key from 4th leaf from 3rd layer. The leaf now has only 29 left in there which is less than half full. Then we move key 13 from (3rd layer/ 3rd leaf) to left and then, moving key 29 to the leaf left of it then we will delete the empty ~~half~~ leaf.

Then the 2nd layer's 2nd leaf we delete key 23 and insert ~~and~~ pointer to leaf with keys (17, 19, 29).
Subsequently, we change the root key from 13 to 17.

Here's it's representation: