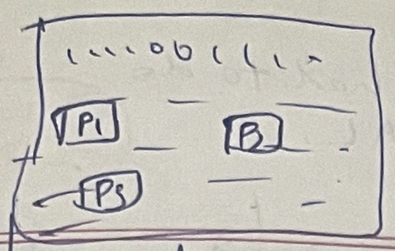
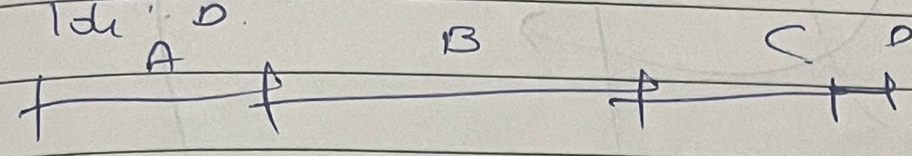


Everything is stored in binaries, how does computer understand what is key, what is value? We have pattern of bits which helps us understand where the key starts, ends, value starts, ends and so on

Iteration 1

101 : A
102 : B
103 : C
104 : D



Search : O(N)
Read : O(N)
Write : O(K) → write K size data by keeping track of end.
Update : O(N) + O(N) = O(N)
read shift.

Binary search not possible as size is not fixed.

Iteration 2 (WAL) written in secondary memory

101 : A
107 : B
111 : C
107 : Z

T1 = 101 : A
T2 = 107 : B
T3 = 111 : C
T4 = 107 : Z (update)
don't do anything abt the existing entry, just write it at the end

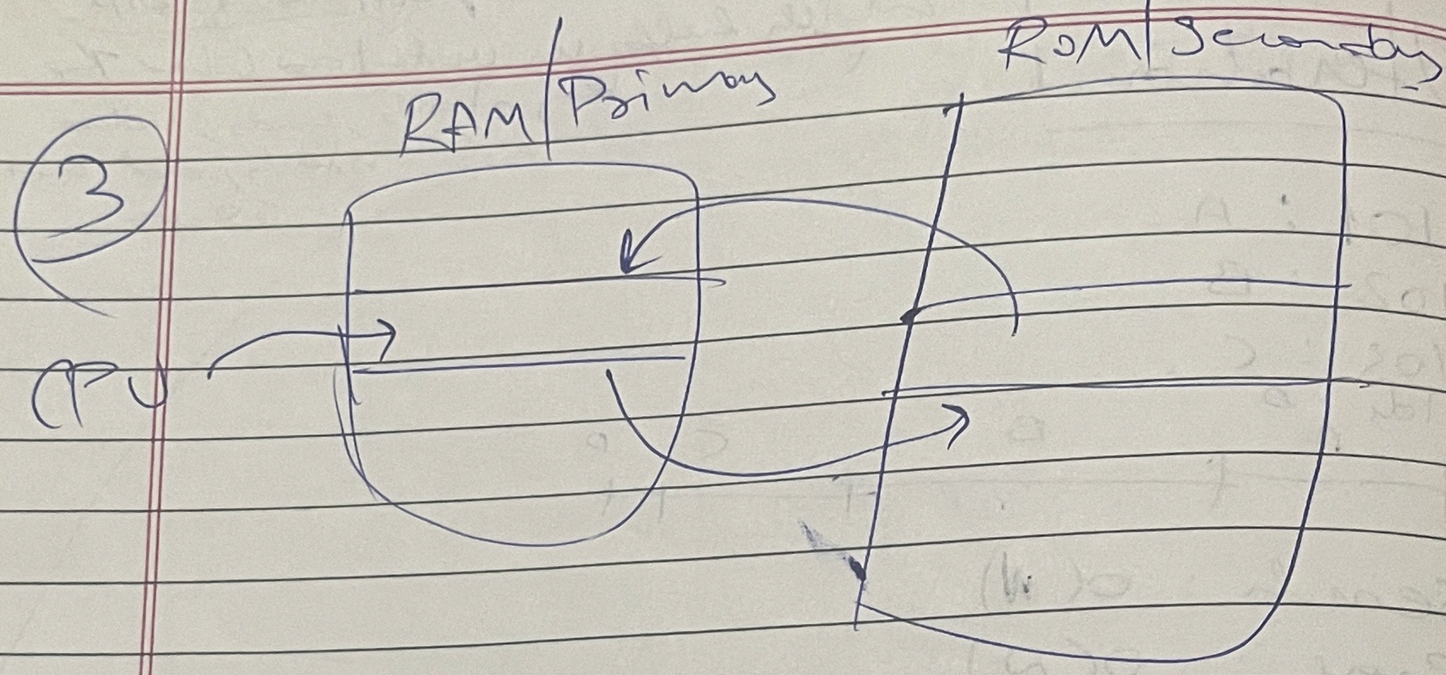
Write : O(1)
Update : O(1)
Read : O(N)
max than N because of duplicates.
Storage space is super bad.

This way: Write & Update O(1) + O(K)

T5 = Read (107), read from back. the first value for key

Delete : O(1)
Mark the key as deleted (update)

Main memory is key-value
Secondary memory → we had to do WAL.



→ If the file size is heavy in secondary memory, the whole file can't be copied to RAM at once.

→ Given that your file which stores your database will keep on becoming bigger and bigger with no time it will become big enough that it can't be loaded to main memory at once

→ File are broken into smaller chunks while storing.

→ If only 100MB of size can be loaded into RAM once why store more than 100MB in one file in secondary memory.

→ As writing to RAM is faster why not keep the new file in RAM until it fills to move it to secondary memory
Solⁿ: ↳ RAM is volatile
↳ keep it in both primary & secondary
File in RAM

MemTable:

Main memory key-value gives us more power as the update could also be O(1) (HashMap)

AOF - Append only File

Iteration - 4

RAM

200 : A
101 : B

HDD

101
102

SS Table(s)

103
111

200 : A
101 : B
200 : Z.

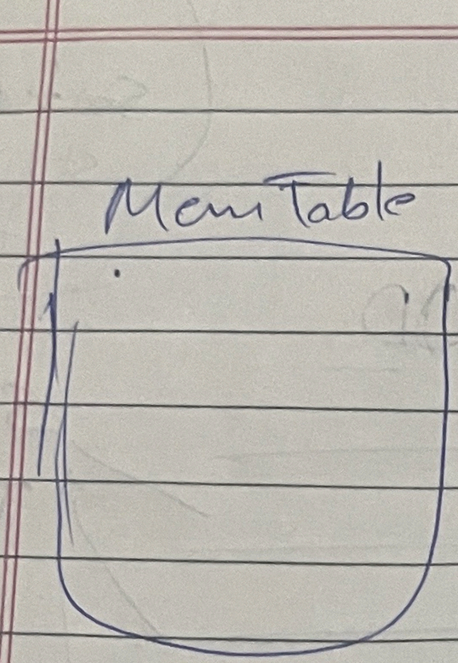
Sorted string table

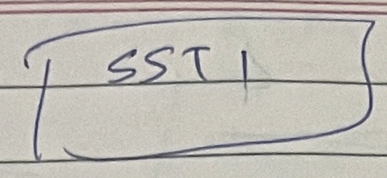
T1 : 200 : A
T2 : 101 : B
T3 : 200 : Z

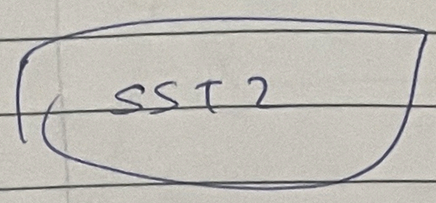
Power goes off → RAM is lost.
→ Chunk 3 is brought from HDD to RAM.
↳ start working again after some time.
delete HDD and get all data from RAM
keep doing this in intervals, advantage we get is HDD chunck 3 will not have duplicates and size would be kept around 100MB
This ensures there will be no duplicates in single chunck, but it can have duplicates from other chunks.

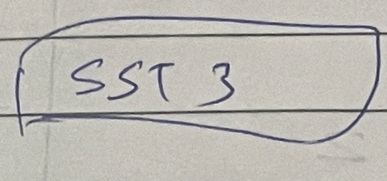
Q Why not have Key-value (HM) as Sorted? ^_^

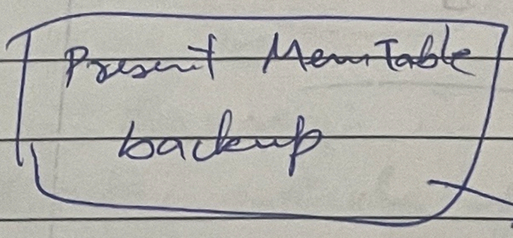
will have HDD in also sorted when synced.











once memTable is filled it will turn into SSTable.

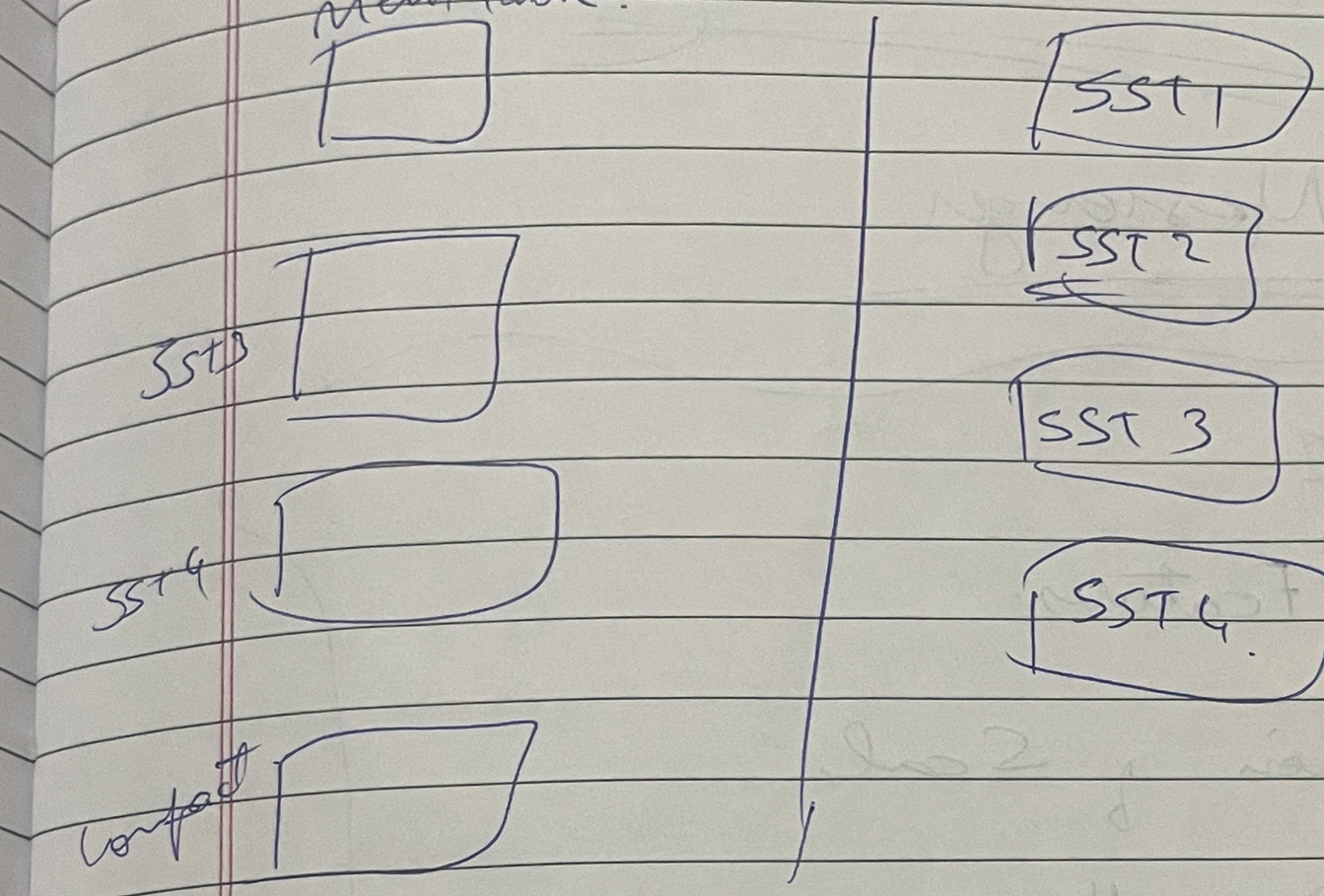
Read (Key)

① Check if MemTable contains the key O(1)

② Look into the latest SST, get back to RAM and read. MemTable remains in RAM.
→ Worst case are still bad.

① Updates → Great
② Write are Great.
③ Reads - Best case Great

Compaction



Iteration 5 : Includs compaction.

Why can't I apply Binary Search?

101 : A
111 : B
200 : C
400 : D