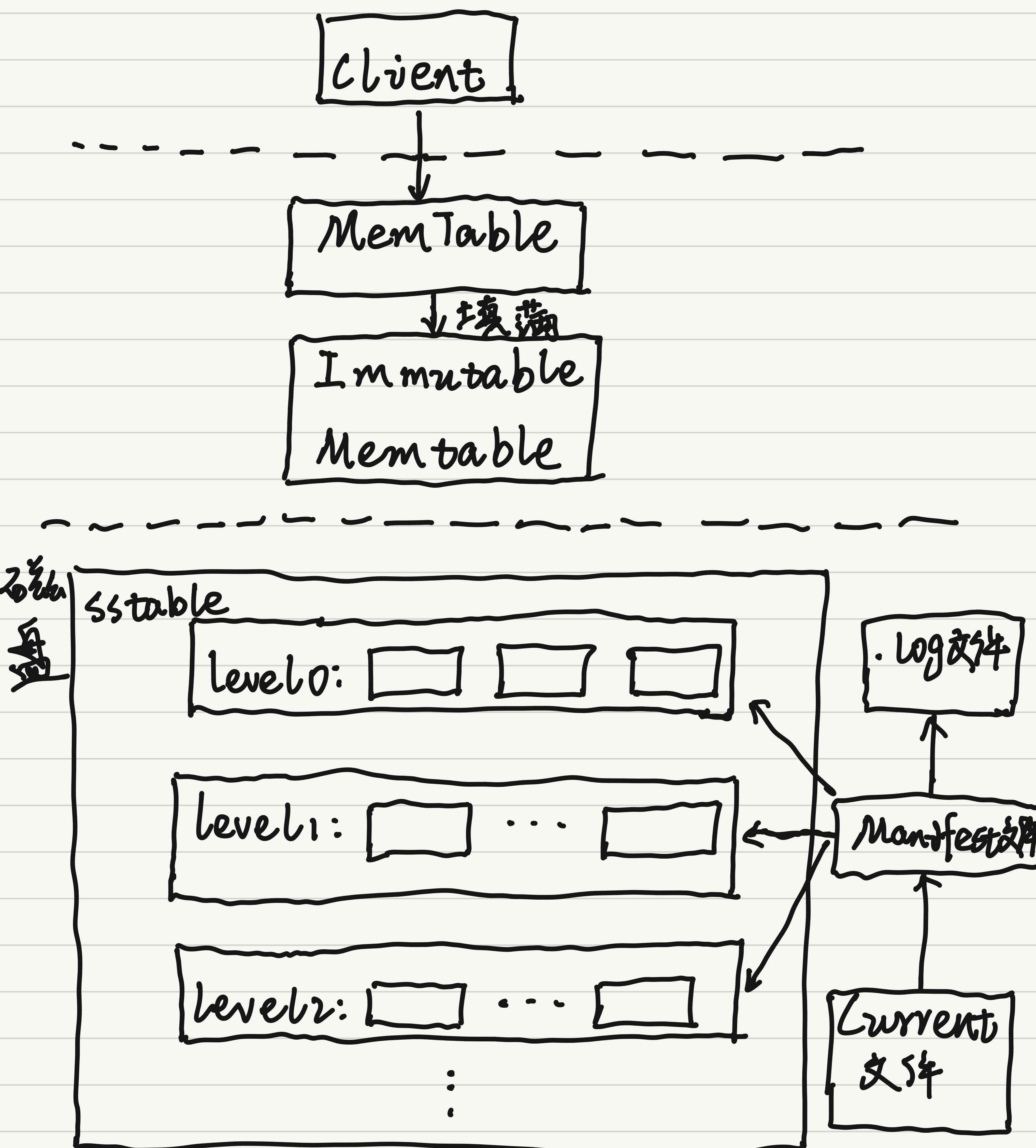


Level db

① 整体架构：



(1). Log文件：以追加写的方式记录日志。

(2). MemTable：内部以跳表的方式实现

(3). Immutable MemTable：当MemTable容量达到

上限后转换而来（只读）

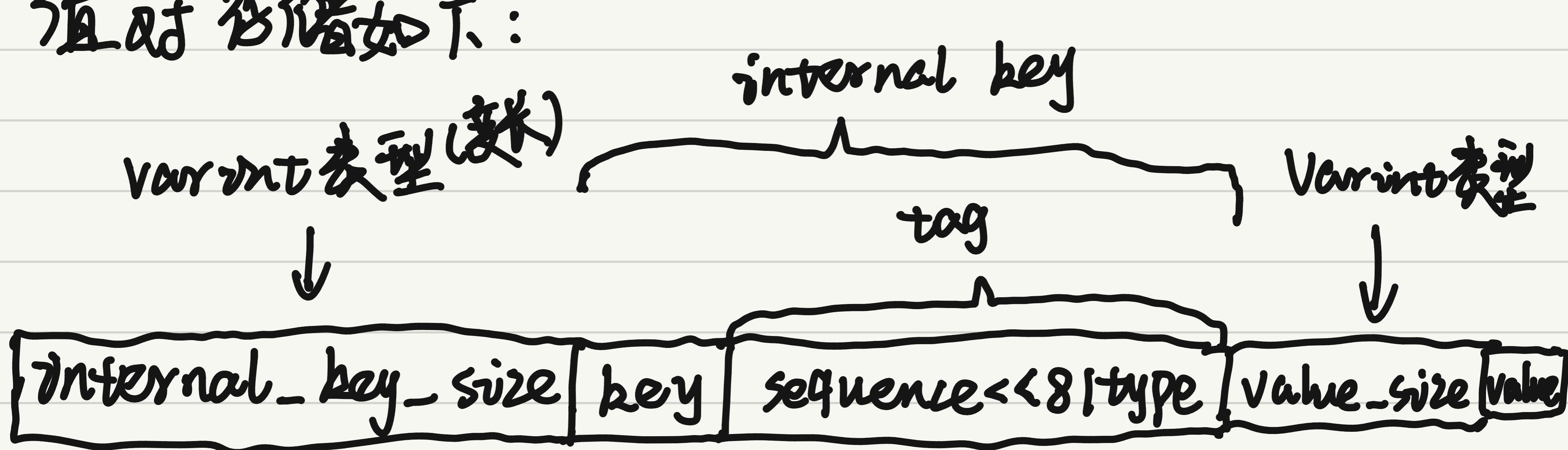
(4). SST文件：层级结构，除level 0层外，其它层级
SST文件均是按键值有序

(5). Manifest文件：记录SST文件在不同level的分布。
以及单个SST文件最大最小key，以及level DB的元
信息

(6). Current文件：记录当前Manifest文件名（因为
不同版本都有其对应的Manifest文件）

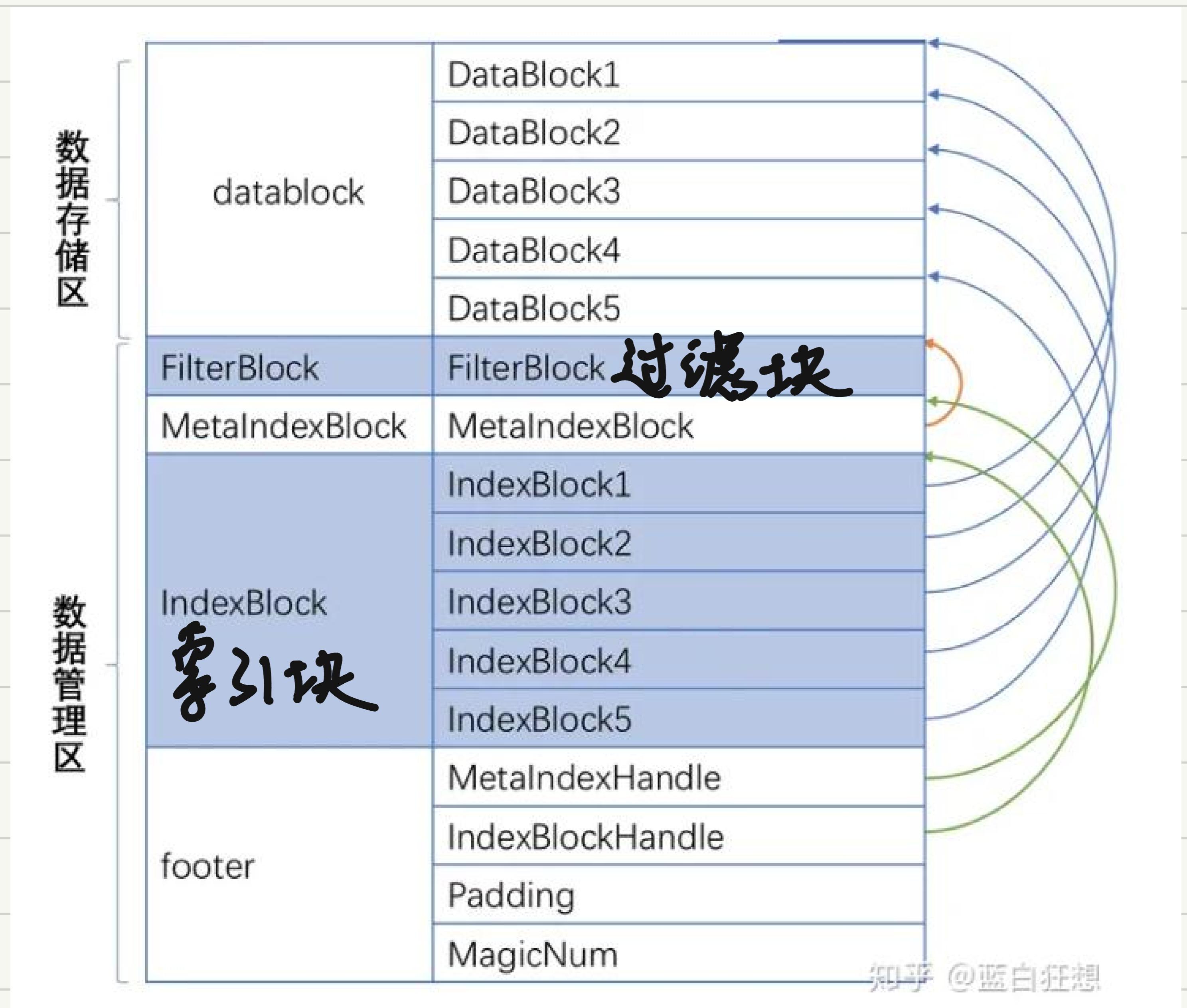
② MemTable 实现:

内部以跳表结构实现，其中跳表的 key-value 键值对存储如下：



其中 `sequence` 记录当前 `key` 的次序，seq 越大，key-value 越新

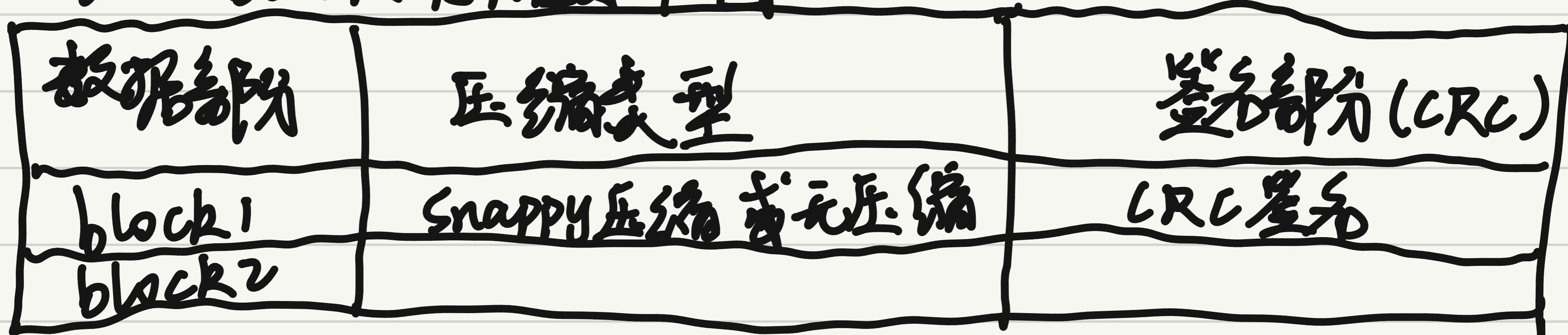
③ SSTable:



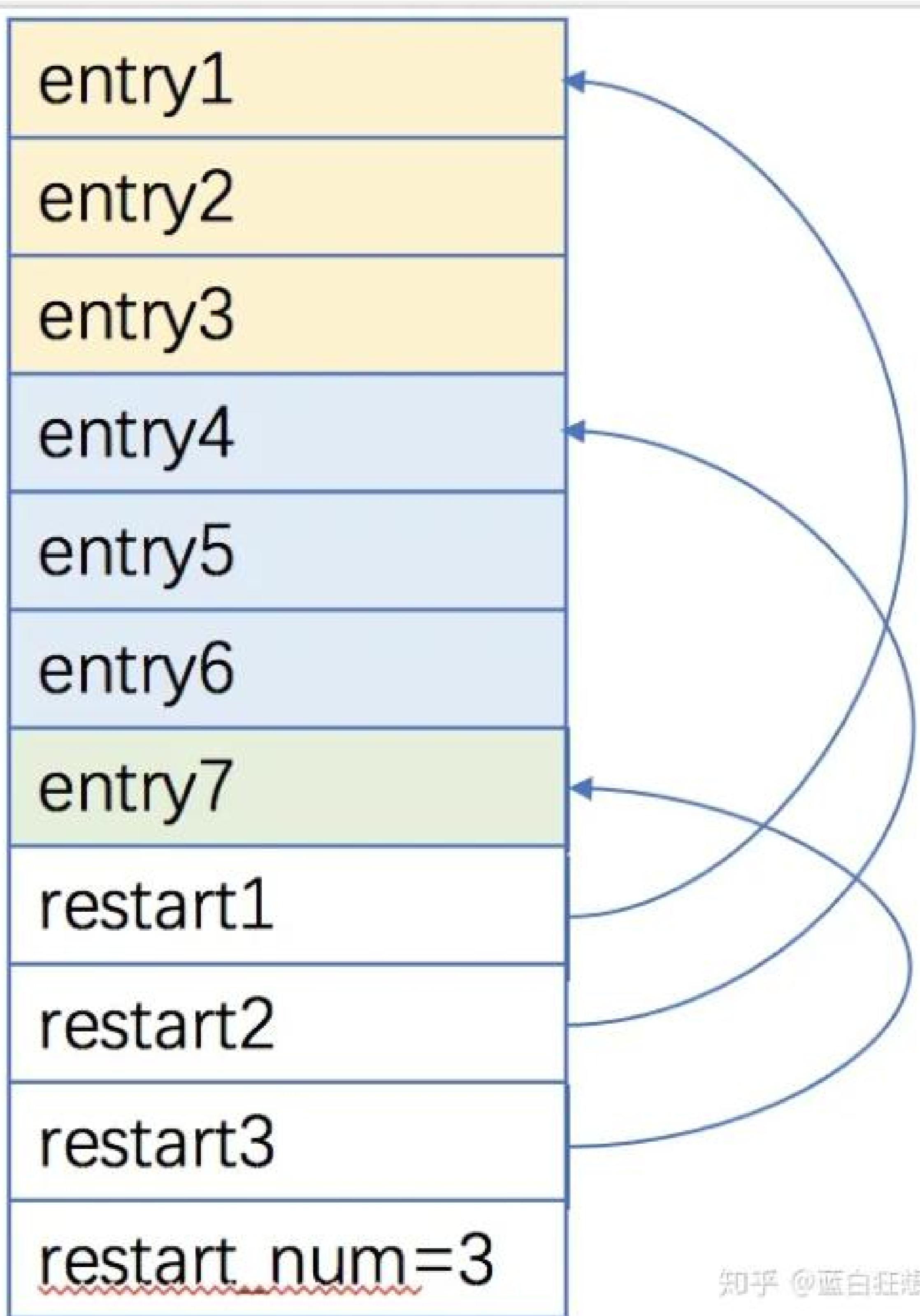
(1). IndexBlock 存储如下图:

key	offset	size
...
...
...

(2). datablock 存储如下图:



其中每个 block 内以 entry 形式存储 - 对 key-value

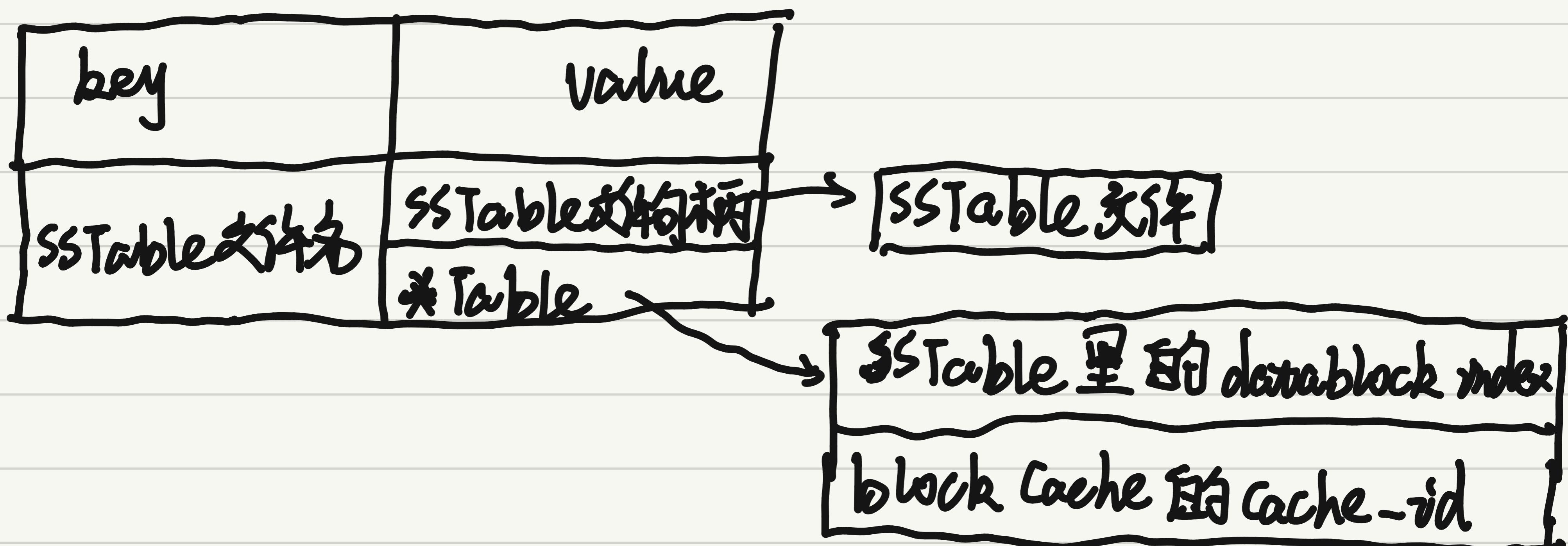


restart 表示重名点，表示两个重名点之间 entry 有一部分
共同 key，便于存储，减少 key 存储量

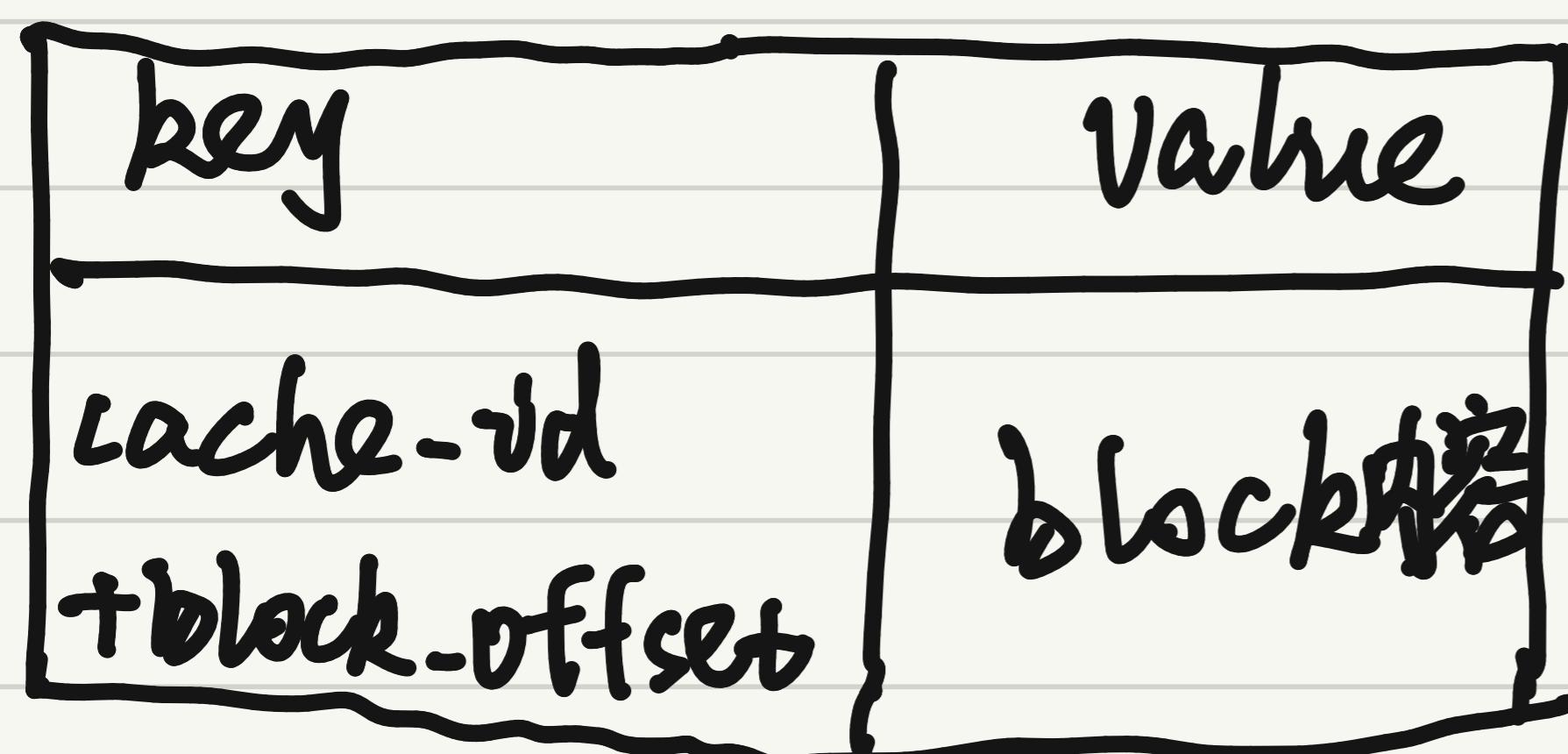
Record i	key共享长度	key非共享长度	value长度	key非共享内容	value内容
Record i+1	key共享长度	key非共享长度	value长度	key非共享内容	value内容

④ Cache (LRU)

(1). Table cache :



(2). block cache:



查找流程：

- (1). 首先根据 SSTable 的 key Range 找到 SSTable 文件。
- (2). 去 Table cache 查找是否有该 SSTable 文件，没有则读入
- (3). 再根据 *Table 找到对应的 Block cache，若没有则通过 SSTable 文件的内部将 datablock 读入 block cache
- (4). 从相应 Block cache 中查找是否有要找的 key，若有则读出，没有则找下一层 Level.

⑤ Version:

- (1) 旧有的文件组成一个版本 version
- (2). 合并操作成为 VersionEdit (生成一个新版本)
- (3). 生成新版本后，旧版本可能仍在服务，就与新版本构成工作
整体
- (4) 旧版本与新版本可以共存，新版本为 Current
- (5). 当旧版本不再服务时，就会丢弃

⑥ 读流程：

(1). 先找 MemTable

(2). 再找 Immutable MemTable

(3). 最后按层级去查找 SSTable

注意：level 0 可能有多个 SSTable 包含要查找的 key，

这时要比较 key-value 键值对的 seq，seq 越大。

key - value 越新

⑦ 写流程：

(1). 先写 log 文件

(2). 再写 MemTable

(3) 若 MemTable 超过容量上限，则转换为 Immutable
MemTable，并合并到 SSTable 中

⑧ compaction:

(1). Minor compaction:

当 MemTable 达到容量上限后，转换为 Immutable MemTable，并与内存中其它 Immutable MemTable 合并成一个 SSTable 存到磁盘中

(2). Major compaction:

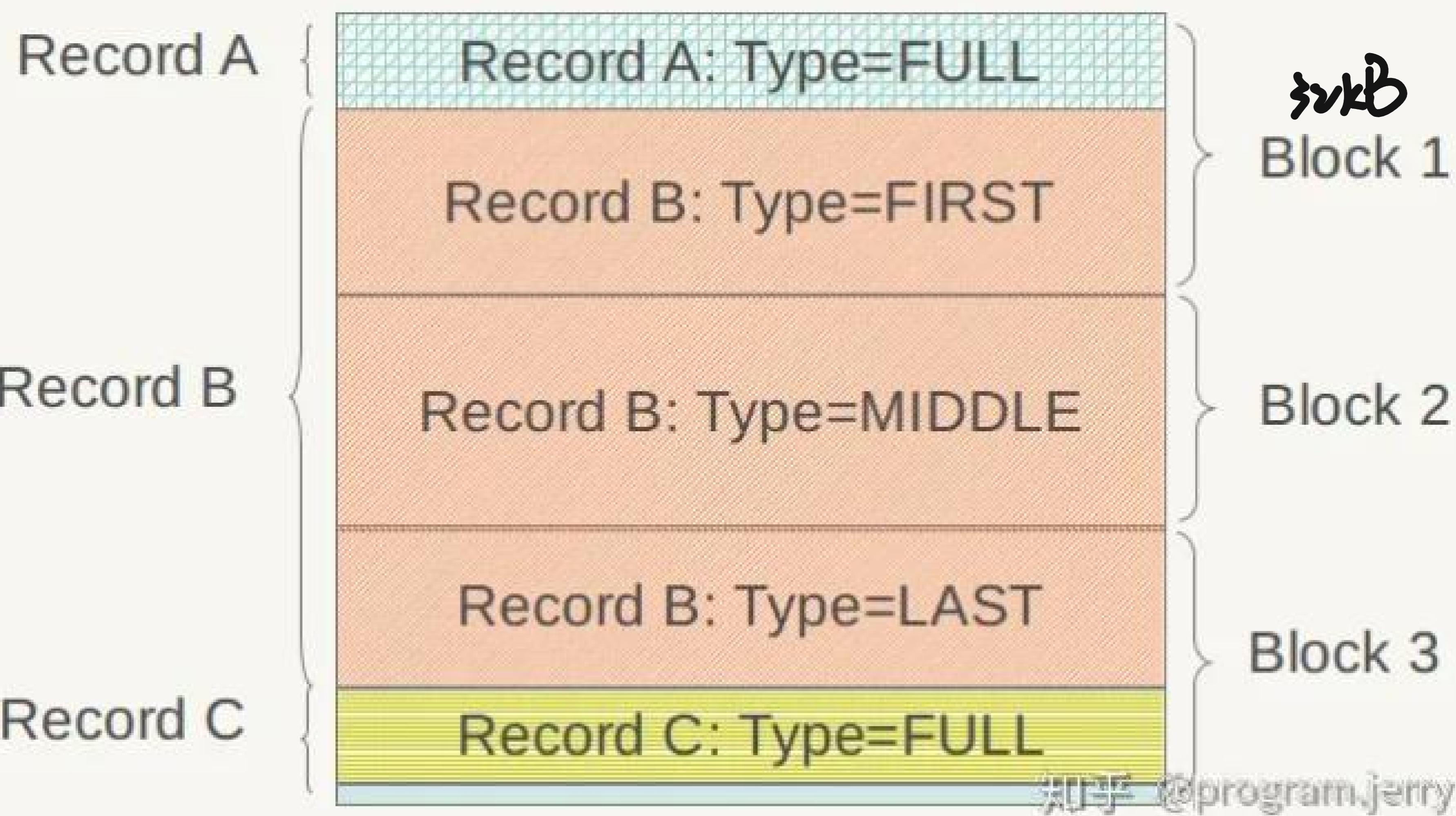
1. size compaction: 某个 level 层级的 SSTable 文件数量超过一定数量将合并到下一层级

2. Seek compaction: 每个新 SSTable 文件中，都包含一个 allowed-seek 的初始阈值，表示最多容忍的 seek miss 次数。每次 seek 失败，该值都会减 1，当减到 0 时，则会合并

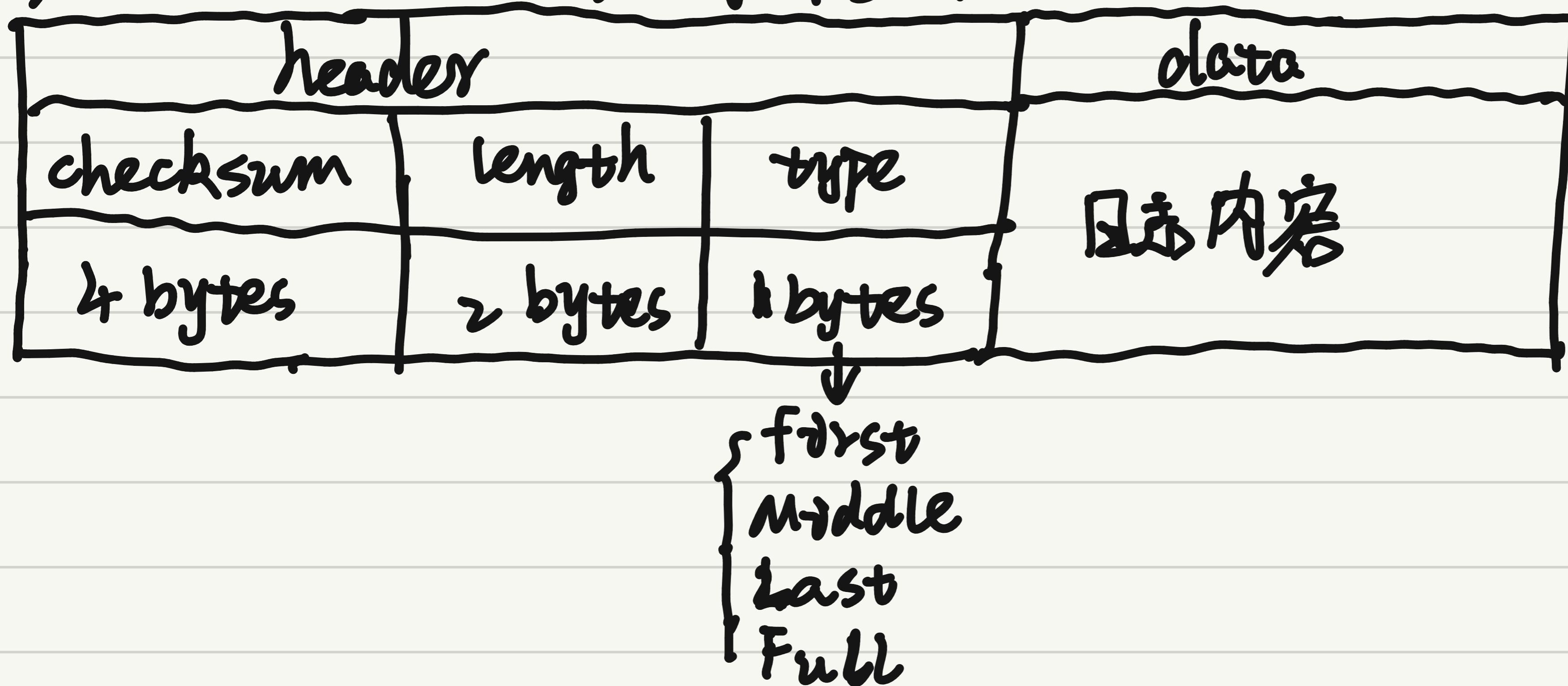
注意：size compaction 优先级大于 seek compaction。

⑨ Log 日志

(1). log 日志结构:



其中 Record 记录结构体如下:



(2). 写日志

若 block 未满且剩余空间 $\geq 7 \text{ bytes}$, 则 追加写

记录头 header (刚好等于 7bytes, 则只写记录头, 封数据长度为 0)

若 block 未满且剩余空间 $< 7 \text{ bytes}$, 则 写到一个新块
且用零填充剩余空间不足 7bytes 的 block

(3). 读日志,

按一个块为单位进行读取日志, 并从块中完整的取出

一个 Record