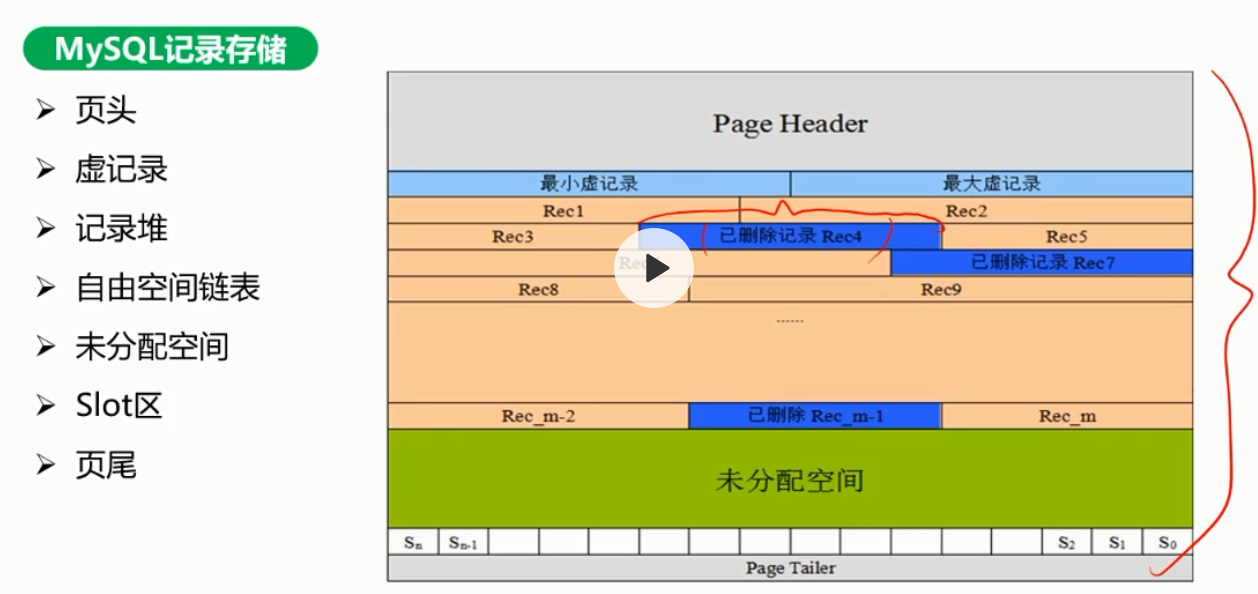
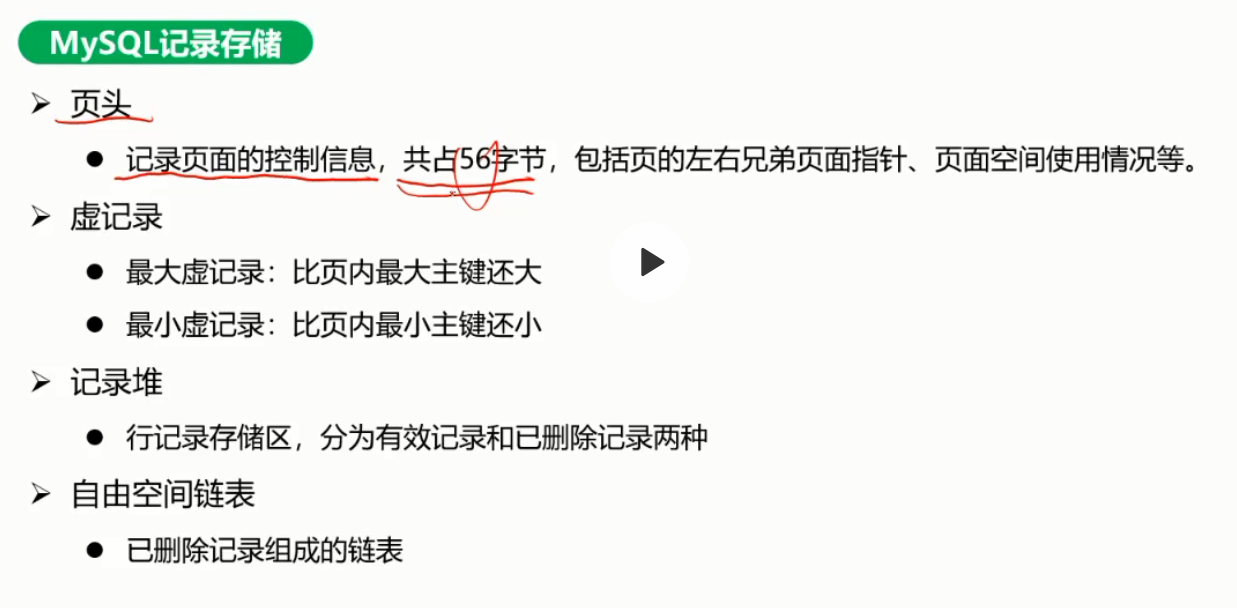
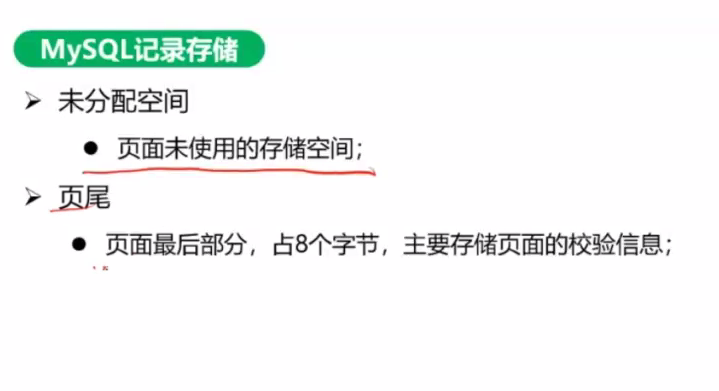
1 页的概念

1.1 Mysql先分配一页，当不够用的时候，再继续分配另外一页。可以认为是mysql最小的管理单位 是16kb内存/页.

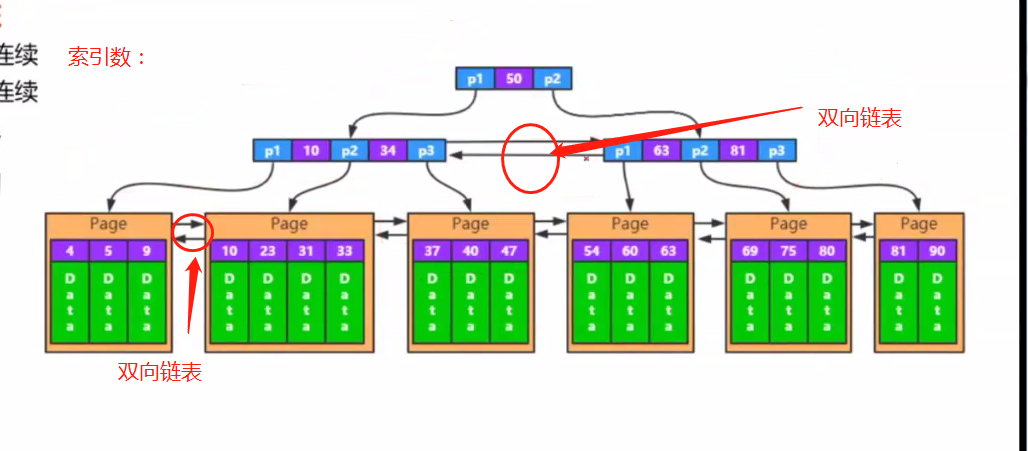
1.2 数据项是存在于最小虚记录和最大虚记录之间的记录项。



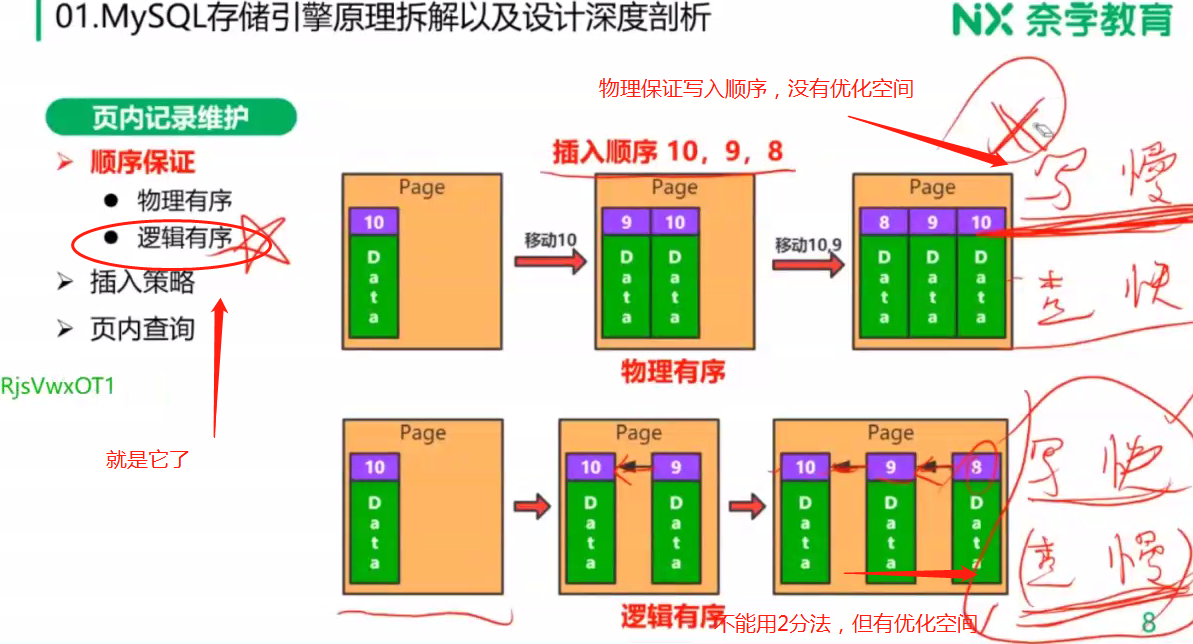




2 业内记录维护的3个点



2.1 顺序保证(如何保证)：



采取逻辑有序，就是链表的形式

2.2 插入策略（插入自由空间链表还是未使用空间？）：

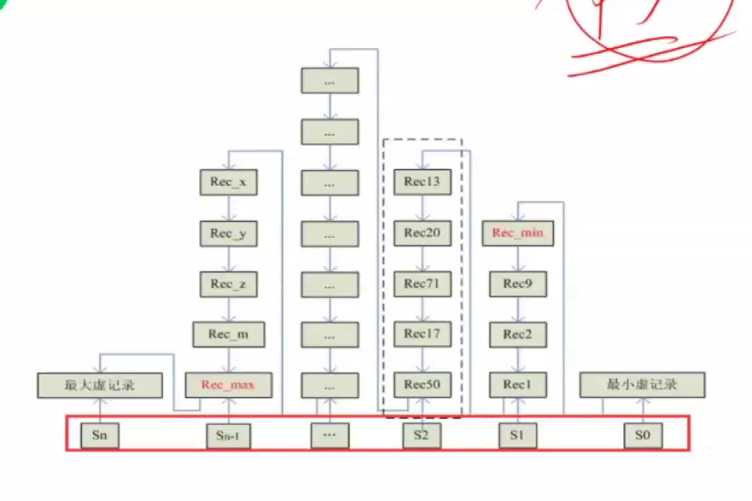
插入到已删除记录（自由空间链表） 还是 插入未分配空间（未使用空间）？

采取先插入已删除记录，后插入未分配空间(mongodb 早期是直接插入未分配空间)

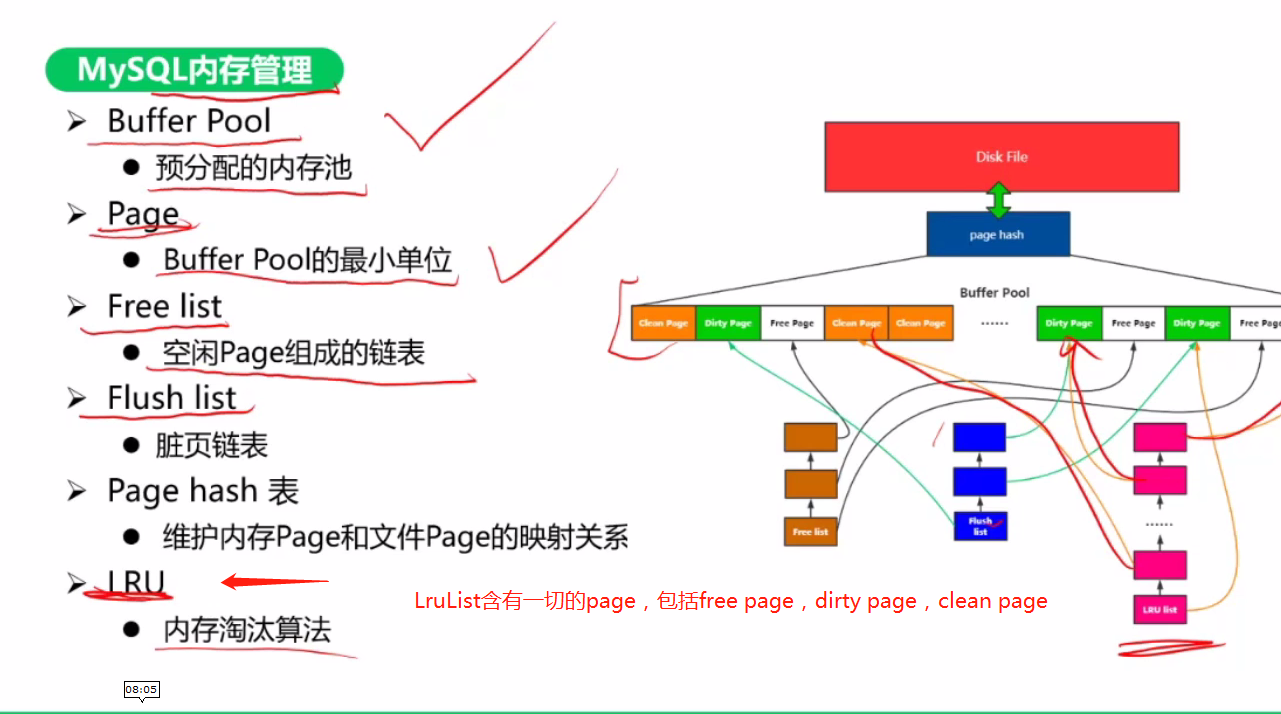
数据库优化方法（写入内存保持紧凑），先删从库，然后从库一条条在从主库拉数据，然后删除主库，把从库转为主库，最后再一条条记录生成他的从库.

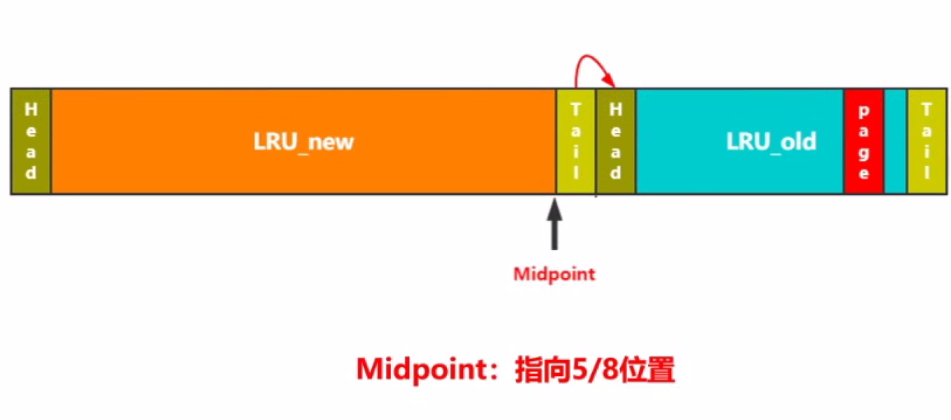
2.3 页内查询：

通过（跳表）二分查询slot区(等值，数组结构，可以快速进行2分)



3 innodb内存管理





3.1 lrulist的操作

3.1.1 页面装载(磁盘到free page)

3.1.2 页面淘汰（free page不足的时候）

从lru list 尾部找到一个dirty page，并写回辞盘，然后这个page成为free page，由free list管理，最后再执行页面加载，放回lru list的old 区间。

3.1.3 位置移动

从old 到new

Innodb\_old\_blocks\_time，当访问一个page的时候，判断下冷数据存活时间是否大于设定的数值，大于才会往热数据迁移

从new到old

移动midpoint，始终=保持热冷比为5:3

3.1.4 lru new 链表的操作

热区，链表结构，非常简便，只需改变3个指向就可以实现把一个热数据给排到最前头。但因为有锁的结构，所以没必要每一次访问都要移动new 链表的。

因此还有优化空间，优化的思路是减少上述的移动次数.

Freed page clock bufferpool的内存淘汰页面数,是全局计数器

Lru new的长度的4/1

当访问这个数据的时候，判断是否需要移动,是根据以下公式来的:

当前freed page clock的计数 减去 这个page上次移动到header时候的freed page clock 要大于 Lru new的长度的4/1 才会发生移动