为什么不建议innodb使用亿级大表

那么为啥不建议用千万级和亿级的大表:

1. 维护的不便利，alter一下...
2. 计算 sum、count过于集中
3. 索引及表数据都是放在 innodb_buffer_pool里面, 数据区间太大，读写热点不交集，造成命中率下降

百万数据 和 亿数据可能B+tree 都 需要三层tree， 但因为百万千万数据的索引空间少，可以更多的放到内存中，速度也就想应的快。 亿表 只能放很小的一部分，万一不中缓存，那么就要走更多的磁盘io 。 innodb_buffer_pool 会缓存前两个层级的B+tree ，这样能更好的更大的存放索引.

对的，说白了主要就是innodb_buffer_pool缓存不够引起的原因 !!!!

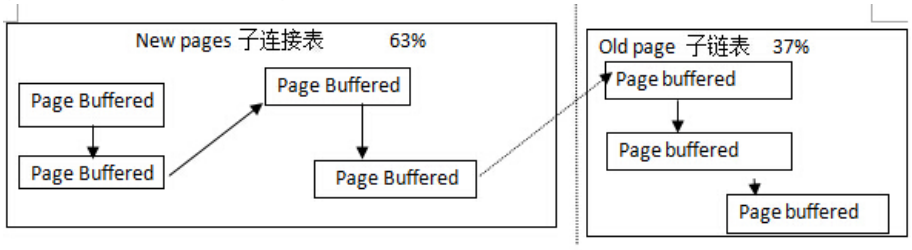
我们来分析下 innodb_buffer_pool的缓存结构及大表带来的缓存污染问题 ？

我们一般说的mysql缓存是 innodb_buffer_pool， 是InnoDB在内存中维护一个缓存池用于缓存数据和索引。（不仅仅是数据，而且有索引， 其实最主要还是B+tree索引！）

缓存池可以被认为一条长LRU链表，该链表又分为2个子链表，一个子链表存放old pages(里面存放的是长时间未被访问的数据页)，另一个子链接存放new pages（里面存放的是最近被访问的数据页面）。old pages 默认占整个列表大小的37%（这个值对应my.conf 的 innoDB_old_blocks_pct 参数的默认值为37，取值范围是5~95），其余为new pages占用。

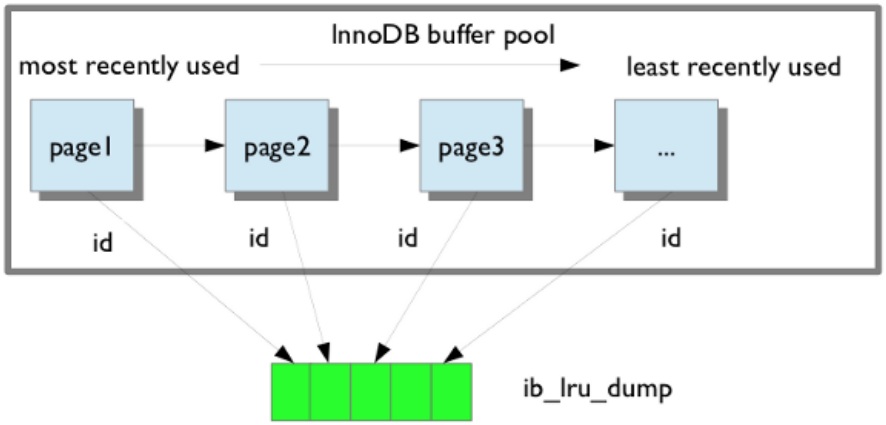
如图下图所示。靠近LRU链表头部的数据页表示最近被访问，靠近LRU链表尾部的数据页表示长时间未被访问，而这两个部分交汇处成为midpoint。

(这图好丑, 没找到像样的图 ...)



InnoDB 缓存池 LUR 链表

当用户需要访问数据时，InnoDB首先会在InnoDB缓冲池查找数据，如果缓冲池中没有数据时，InnoDB会将硬盘上的数据块插入到InnoDB缓存池中；如果InnoDB缓冲池已满，InnoDB通过LRU算法清楚InnoDB缓存池中个别数据块。每当有新数据块需要加载到InnoDB缓冲池中时，该数据块应变为 ‘ ‘数据页’ ’ 被插到midpoint的位置，并声明为old数据页。那么old数据页什么时候能移动到new Page链表中呢？



（1）当InnoDB_old_blocks_time的参数值设置为0时。当old部分的数据页被访问到时，该数据页会被提升到链表的头部，并被标记为new数据页。
（2）当InnoDB_old_blocks_time的参数值大于0时（以1000毫秒或者1秒为例）。

old部分数据页插入缓冲池后，1秒之后被访问，该数据页会被提升到链表的头部，并被标记为new数据页。在刚插入到一秒内，即便old部分的数据页被访问，该数据页也不会移动到new链表的头部。

那么所以呢 ？ 表数据那么多，总是冷不丁去查询时老数据，那么这种不频繁的page就会被挤出innodb_buffer_pool之外，使得之后的SQL查询会产生磁盘IO，从而导致响应速度变慢 。
我们当时有个article_keyword表，数据行为20亿左右，时间区间有1年左右，这里时常会查询 比较老的数据和去重判断…. 因为区间实在是够大，动不动就把LRU链表的热数据给挤走了…

再进一步分析下，一个sql会产生多少磁盘io？

这里我们又要引入两个索引概念，一个是聚集索引，一个是非聚集索引… 这两个概念很重要，涉及到了innodb引擎的设计.

- 聚集索引一个表只能有一个，而非聚集索引一个表可以存在多个 ！
- 聚集索引存储记录是物理上连续存在，而非聚集索引是逻辑上的连续，物理存储并不连续 ！

请大家牢记这个定义，面试的时候会问到哦…

聚集索引只能是一个？ 对的，可以理解为表的主键索引。其他的单独索引及联合索引可以理解为非聚集索引.

如果你没有自己指定主键的化. innodb 会自动创建一个主键. 主键就是聚集索引，也只有主键才能聚集索引, 那么你可能不需要主键，innodb创建主键的意义在于什么？ 让非聚集索引拿到数据…

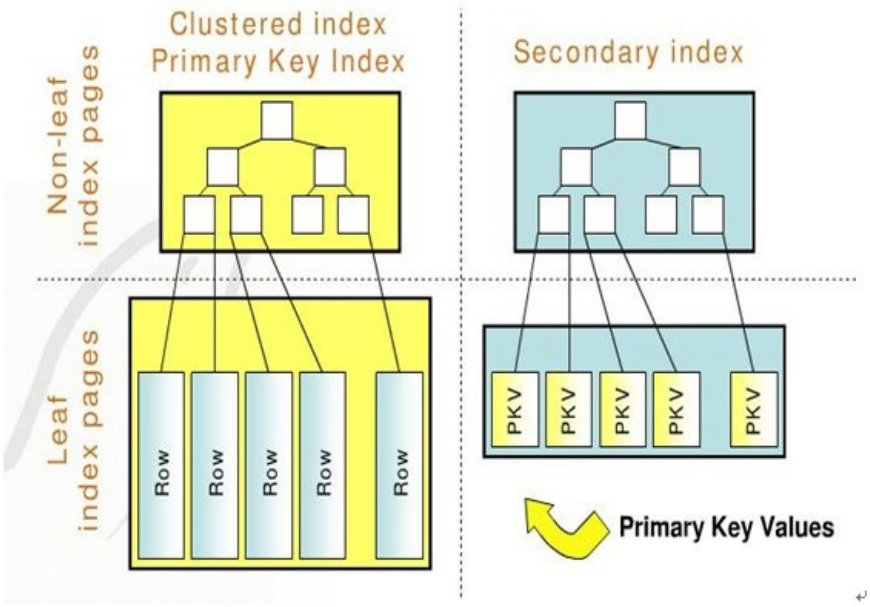
聚集索引和非聚集索引的索引走向：
首先通过innodb的查询优化器判断你的请求是否是聚集索引请求。
如果是< 聚集索引 >，那么会在 innodb_buffer_pool里找到第一层B+tree, 如果找到区间，那么继续找第二层B+tree， 最终拿到row 数据 。
如果是< 非聚集索引 >，那么就是找到相关的非聚集索引了，通过索引字段查到对应的主键，然后拿着主键去拿聚集索引拿数据。
机械硬盘一般都在每秒200以下的iops，姑且按照200iops来计算，那么一次io差不多是在 5ms 左右，那么在没有命中任何索引的情况下，你要花费多少时间呢？ 自己推到公式吧.

1	7200rpm的磁盘IOPS=1000/(9+4.17)=76IOPS
2	10000rpm的磁盘IOPS=1000/(6+3)=111IOPS
3	15000rpm的磁盘IOPS=1000/(4+2)=166IOPS
4	构建raid1orraid10磁盘阵列可实现成倍的增加iops

固态硬盘的iops能到多少？ 一般达到随机3万不是问题.. 那么单次iops也就几十微妙吧，具体时间你可以推算下，
最坏最慢的结果是： 通过非聚集索引找到主键，花费3个io，又通过主键找到具体数据，又花费了3个io。 mysql innodb会把前两层的索引装载到内存中的 。

那么你肯定又会问了，innodb B+tree 如果出现 N 层, 那么就走N个IO吧？ 层数很大怎么办？
这么说吧，100亿的数据也顶多出现4层的B+tree….
参考下面的图片，应该更好理解吧. 左面聚集索引，右面非聚集索引，聚集索引通过B+tree的查询直接拿到row数据，而非聚集索引只能拿到他的主键标记，然后通过主键才能查询到数据。 （这图片是google搜到的）

索引


Clustered index
Primary Key Index
Secondary index
Non-leaf
index pages
Leaf
index pages
Row
Row
Row
Row
Row
PKV
PKV
PKV
PKV
PKV
Primary Key Values