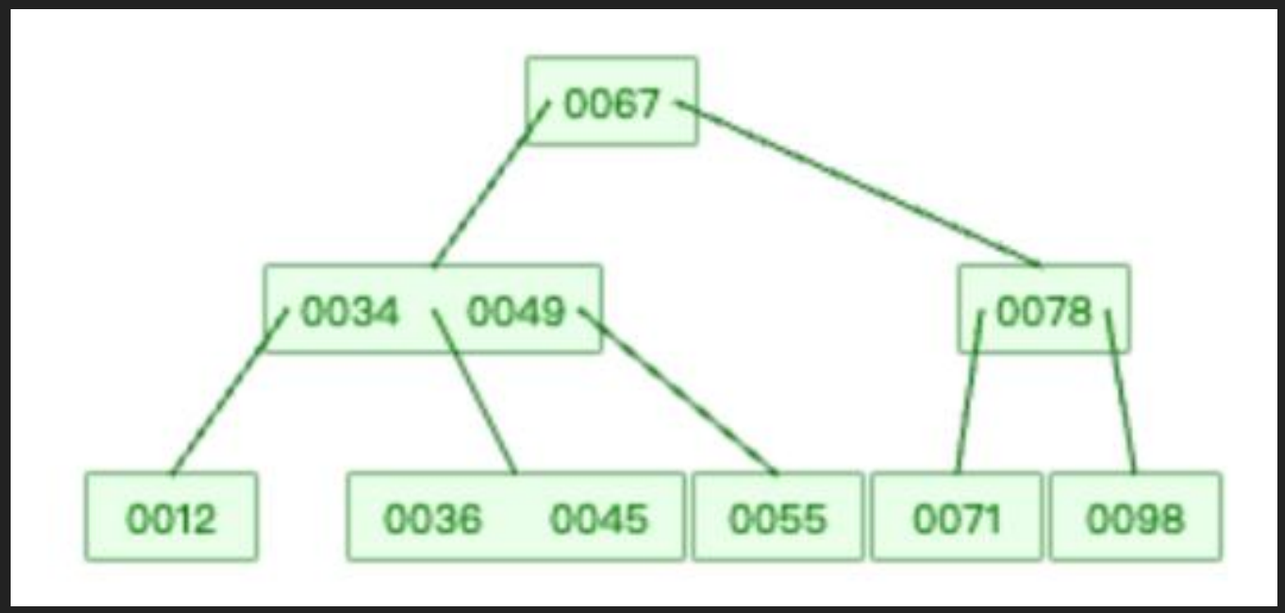
内容大纲

- B树、B+树
- LSM树

B树

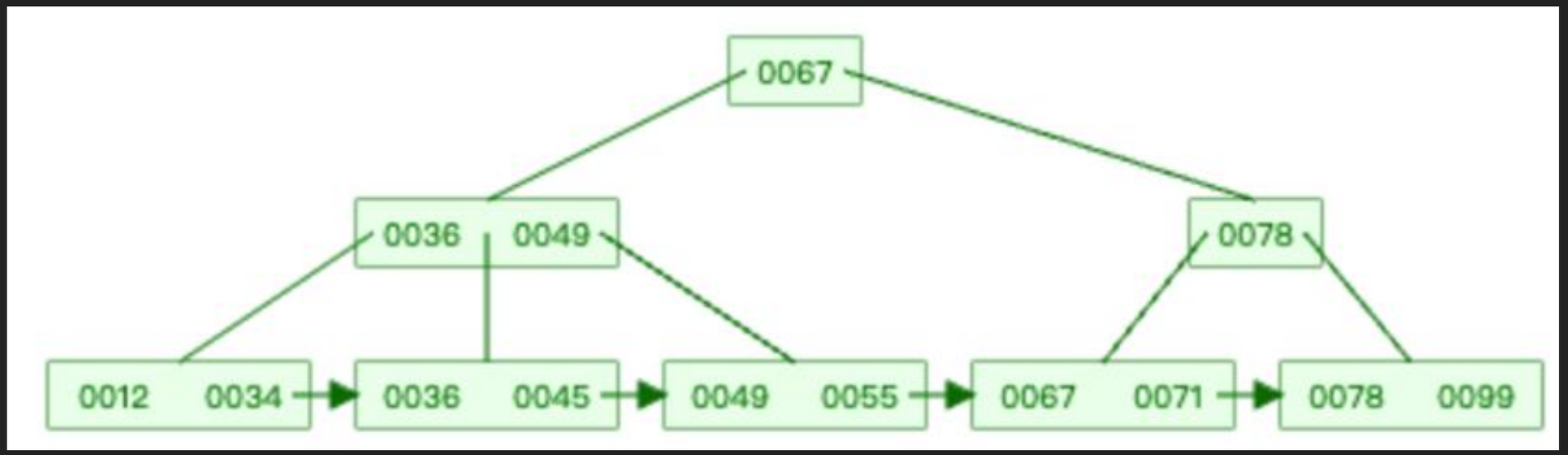
12、34、36、45、49、55、67、71、78、99



- B树每个节点可以有多个子树，M阶B树表示该树每个节点最多有M个子树
  - 若根结点不是终端结点，则至少有2棵子树
  - 除根节点以外的所有非叶结点至少M/2棵子树，至多有M个子树（关键字数为子树减一）
  - 所有的叶子结点都位于同一层

B+树

12、34、36、45、49、55、67、71、78、99



- B+树是B树的一种变体，其与B树的区别如下：
  - 叶子节点（最底部的节点）才会存放实际数据（索引+记录），非叶子节点只会存放索引。
  - 叶子节点按照索引的顺序从小到大连接起来。
  - 所有的非叶子节点包含了一部分的索引数据，节点的索引数据为其子节点中最大或者最小的索引数据。

MySQL 单表数据量大于 2000 W行，性能会明显下降？

在Innodb存储引擎里面，最小存储单元是页，而一个页的大小默认是16KB。一个节点（叶子节点或非叶子节点）的大小就是一页。

假设：

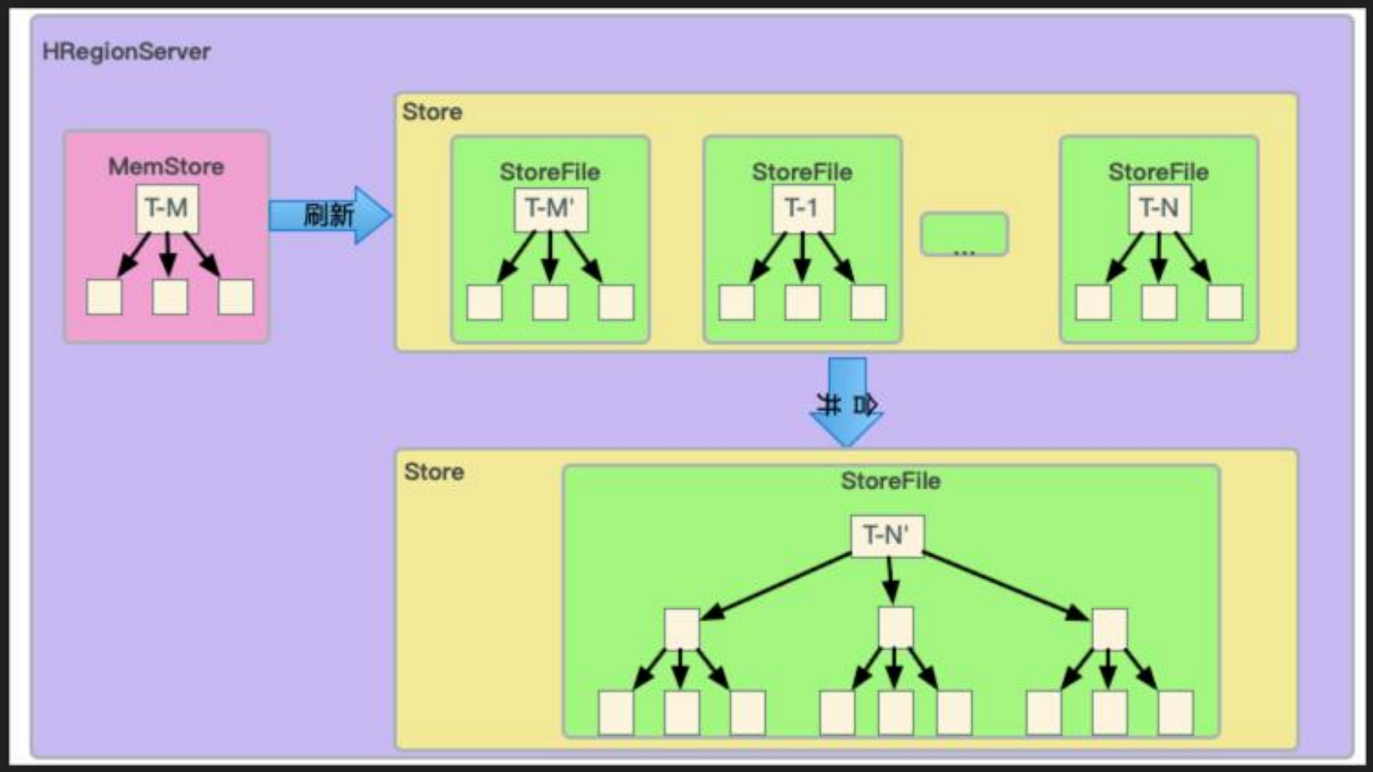
- 主键类型为bigint，占用8Byte，指针可以设置为占用6Byte，总共14Byte。
  =》非叶子节点可以存放 16KByte/14Byte=1170 个"主键+指针"的组合。
- 一行数据大小是1K，一个叶子节点就可以存16KByte/1K=16条（行）数据。

2层B+树的话：可以索引1170个*16条=18720条（行）数据。

3层B+树的话：可以索引1170个*1170个*16条=21902400条（行）数据。

HBase入门与实践

LSM树



- LSM树（log structured merge-trees）核心思想是牺牲一定的读性能来换取写能力的最大化
  - HBase数据的写入都会先写MemStore，在内存中构建一棵有序的小树
  - 当MemStore达到一定条件时即会刷新输出写入到磁盘为一个StoreFile，此时的数据已经有序并且因为是顺序写入磁盘，所以写入速度很快
  - LSM中M代表的merge（合并）就是为了解决StoreFile越来越多而造成读性能下降的问题
  - 当MemStore刷新后StoreFile达到配置的数量或者距离上次压缩时间满足配置的间隔时，HBase即会自动触发压缩