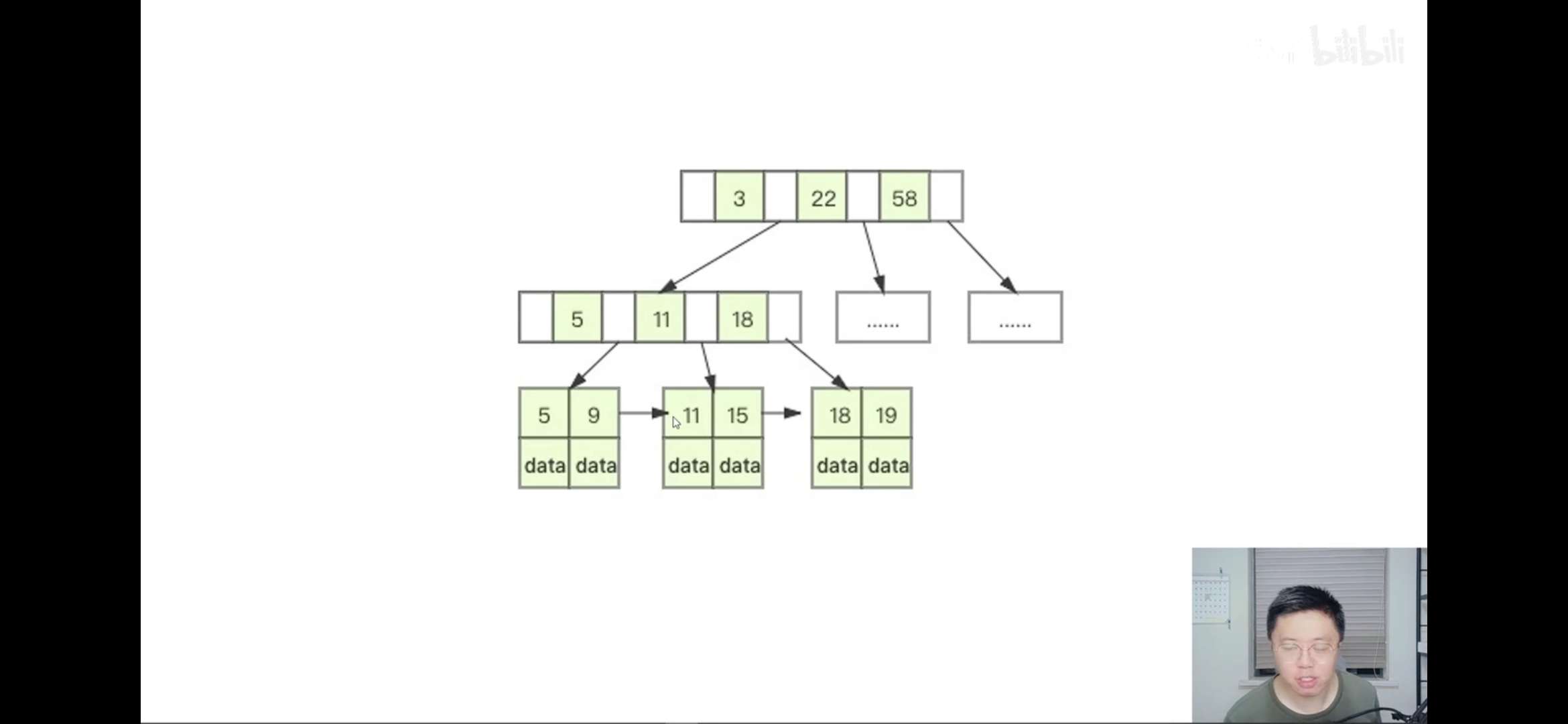
## 索引（uuid导致的生产问题）

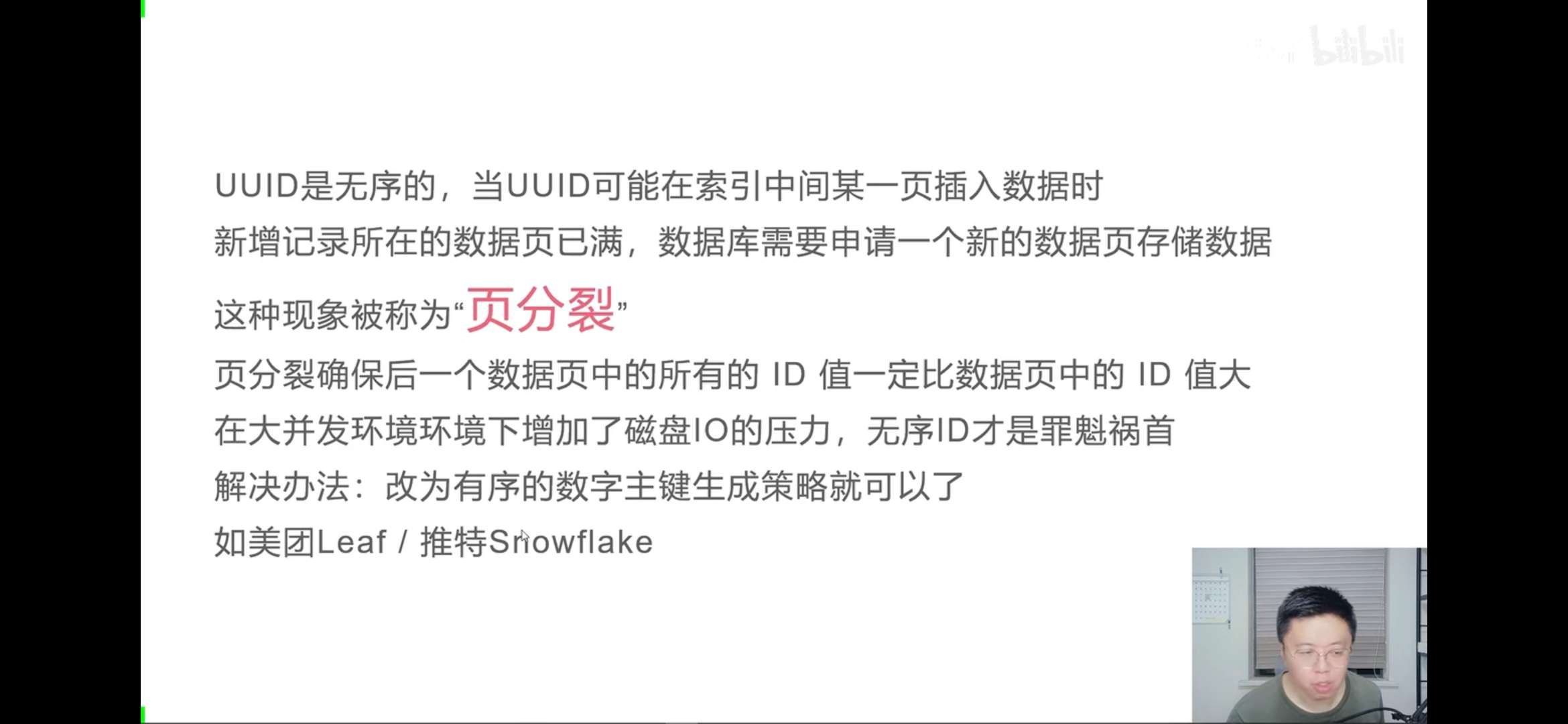
mysql 索性 簇级索引-也就是索引和数据关联很高 b➕ 树

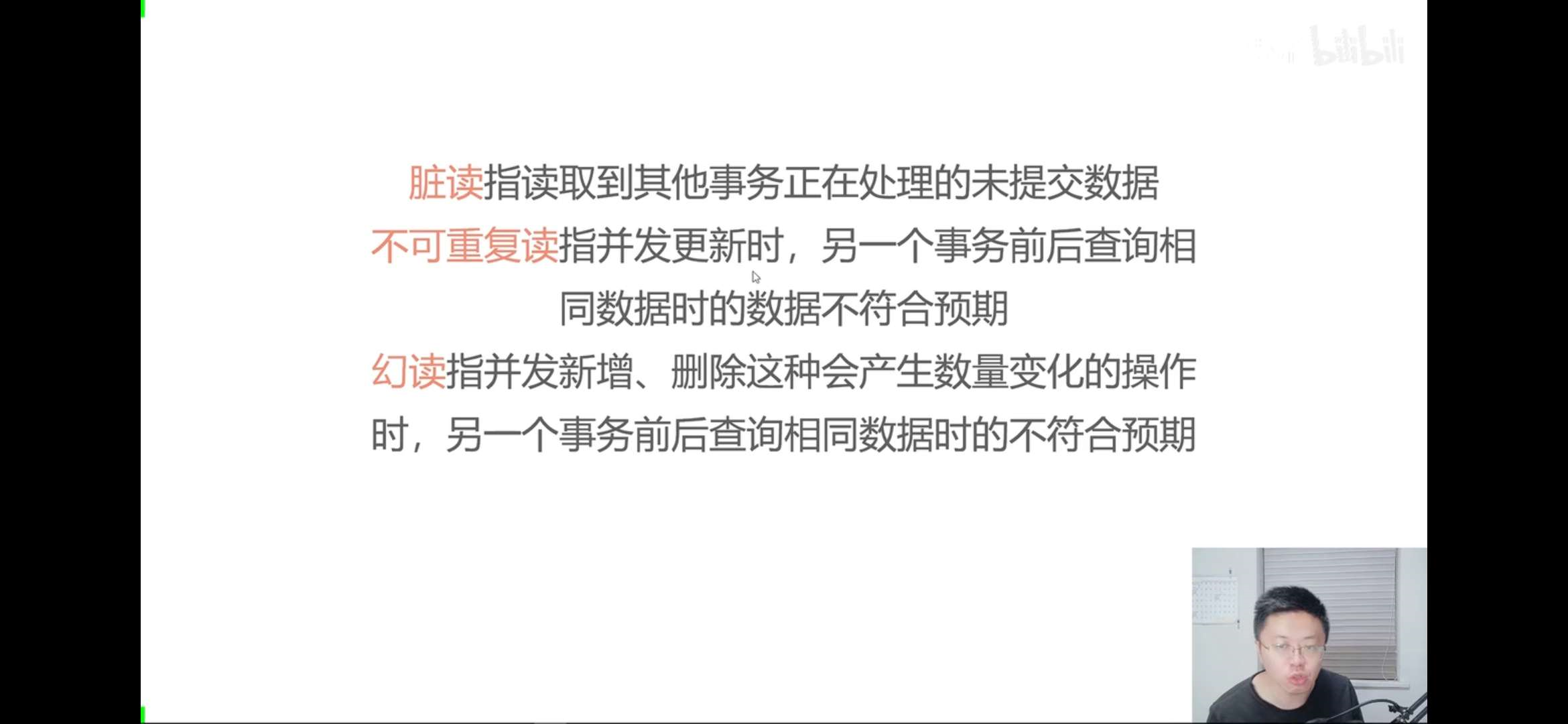


后面的叶子节点比前面的叶子节点值要大

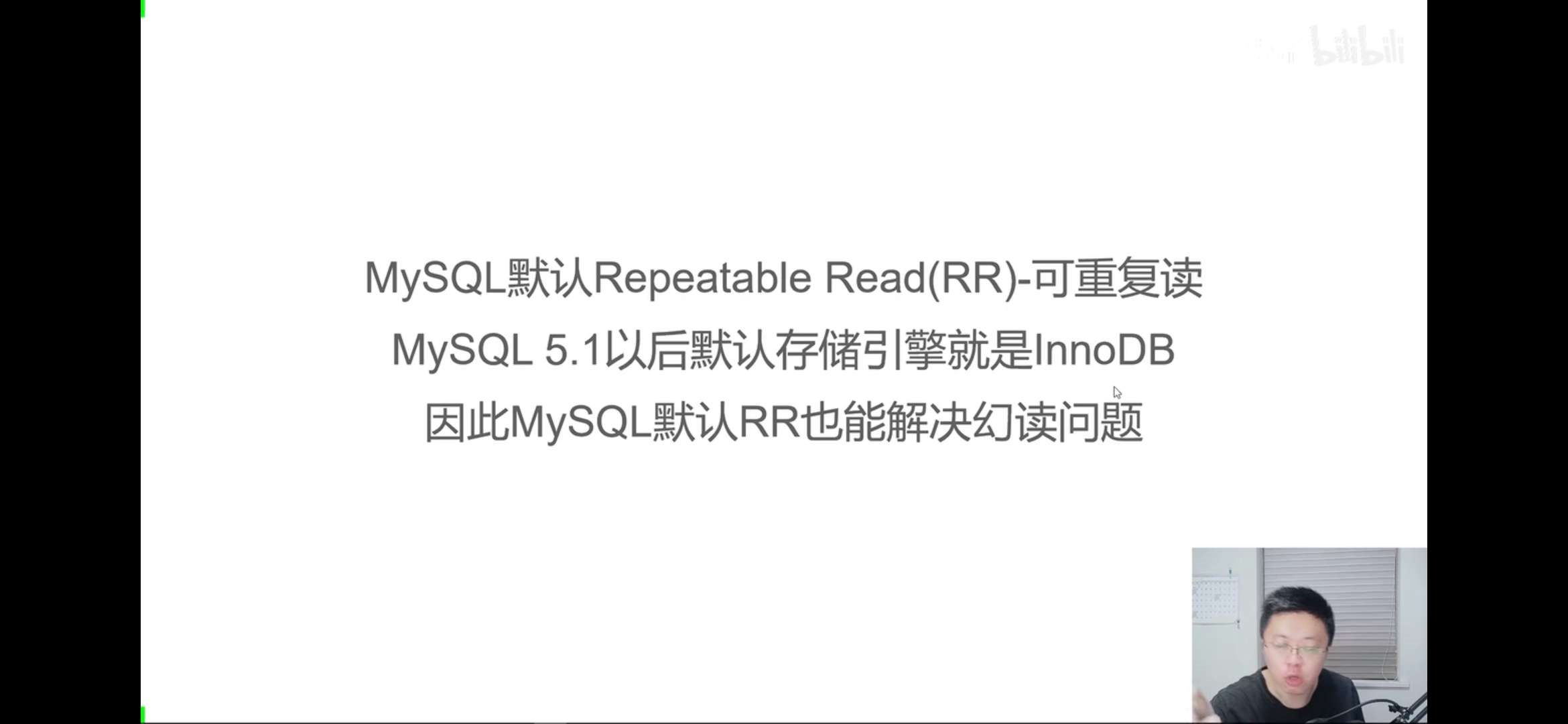
uuid大数据量延迟大 原因 页分裂

把uuid作为主键换为其他的有序主键

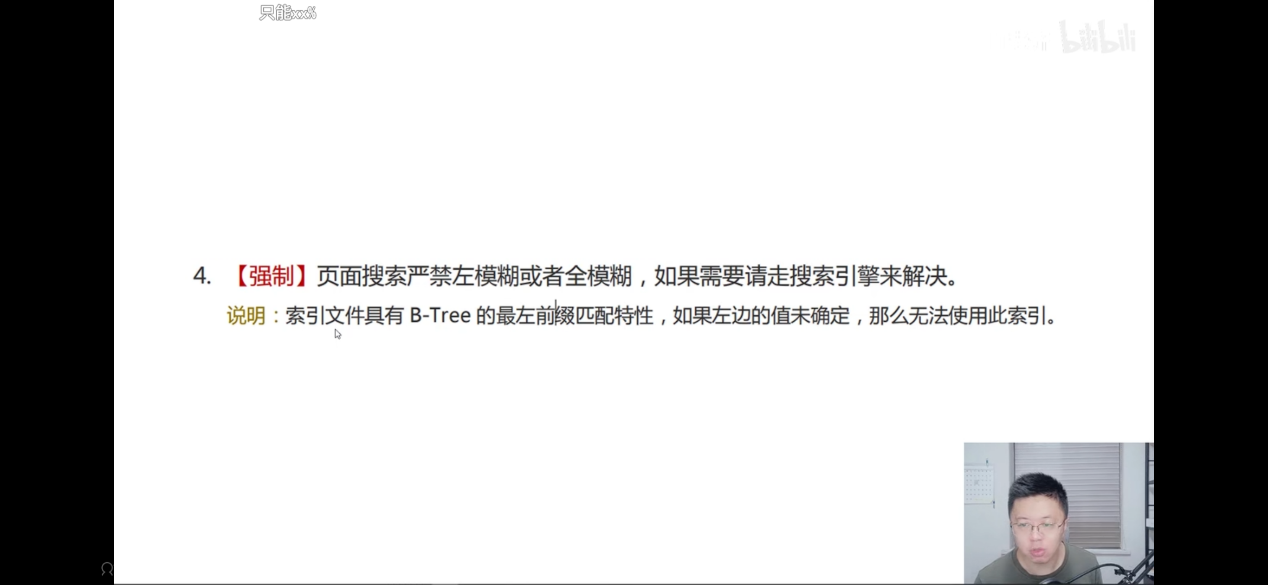


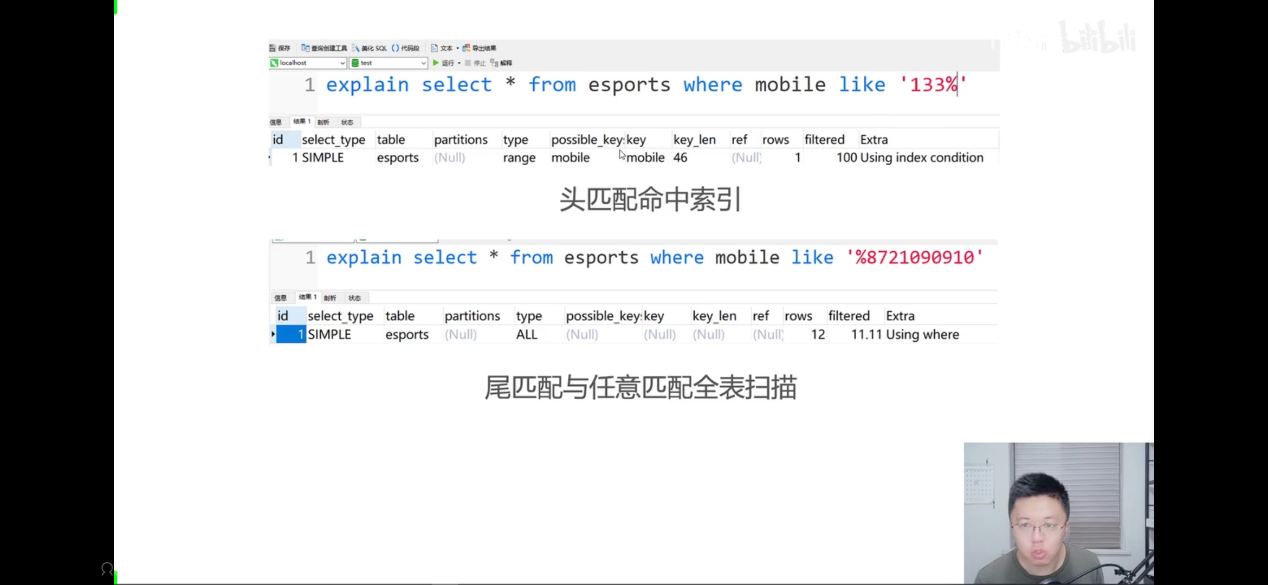




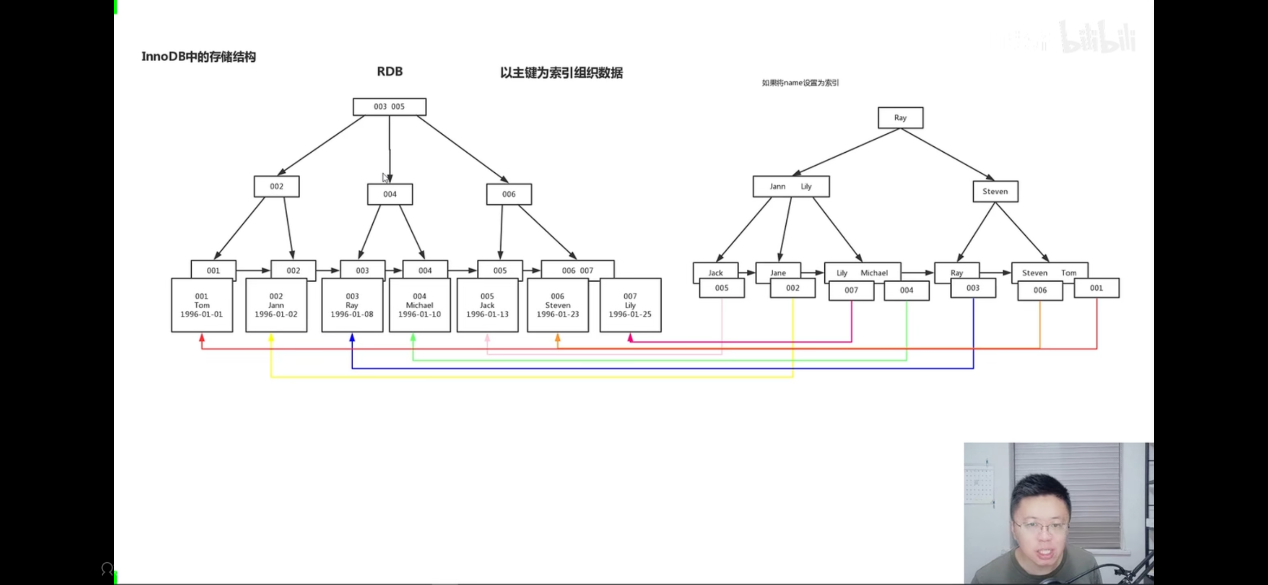


## 模糊匹配





聚簇索引

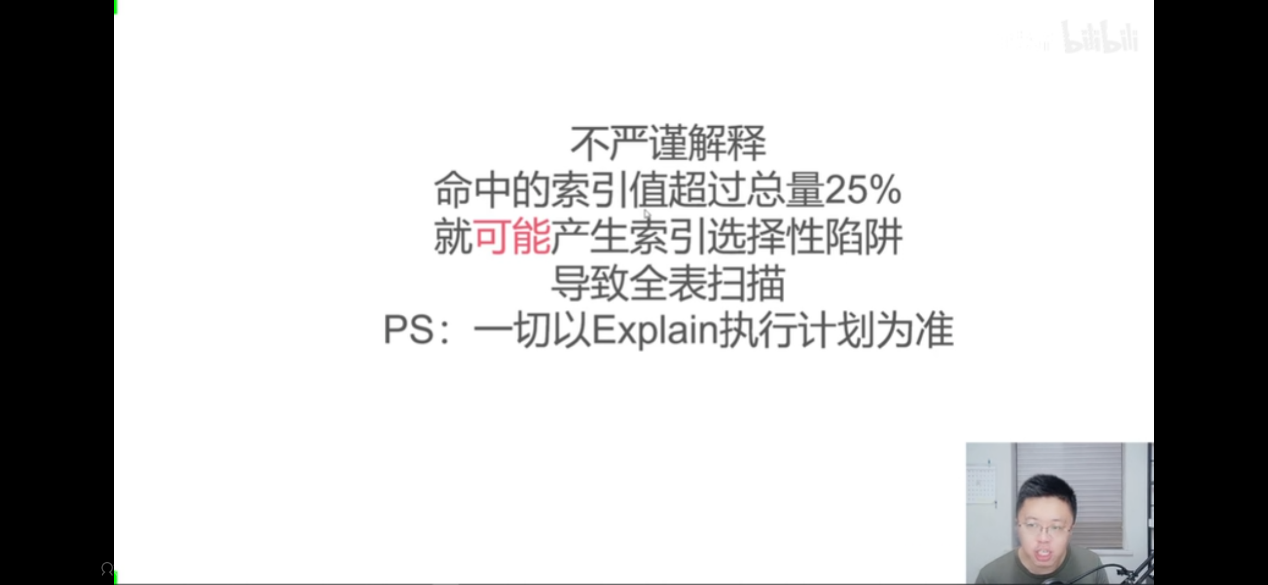


索引的前后顺序和数据的前后顺序是一致的



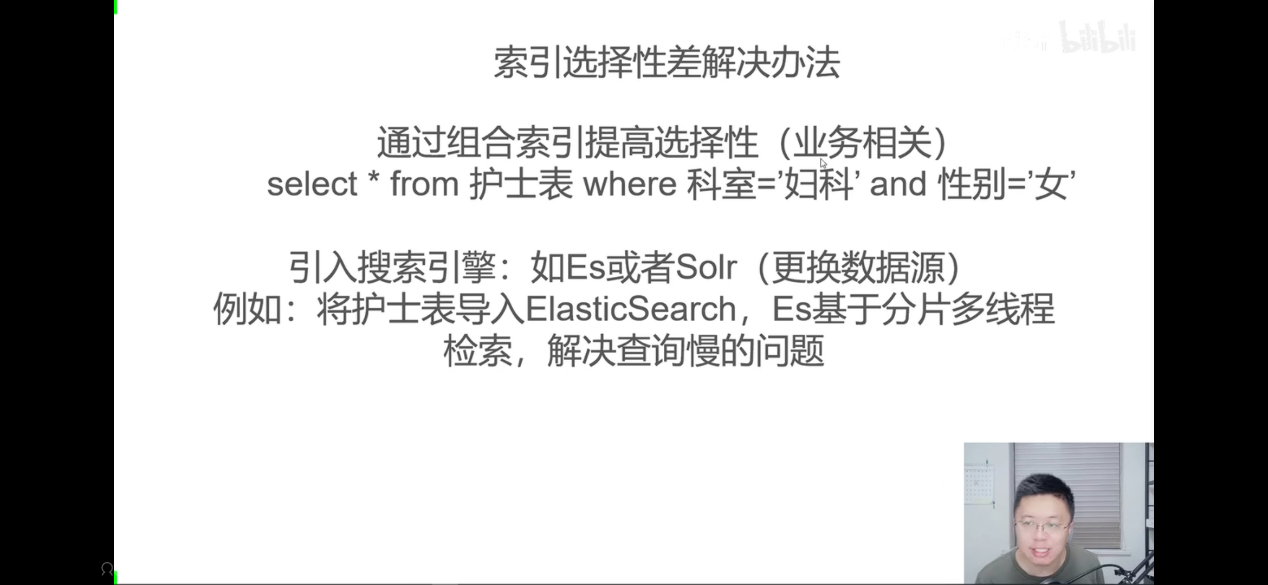
当这个sql 模糊查询已经包含80% 90%的数据时就会不走索性

因为有索引还有个遍历的索引的消耗



索性选择性差就会导致全表扫描

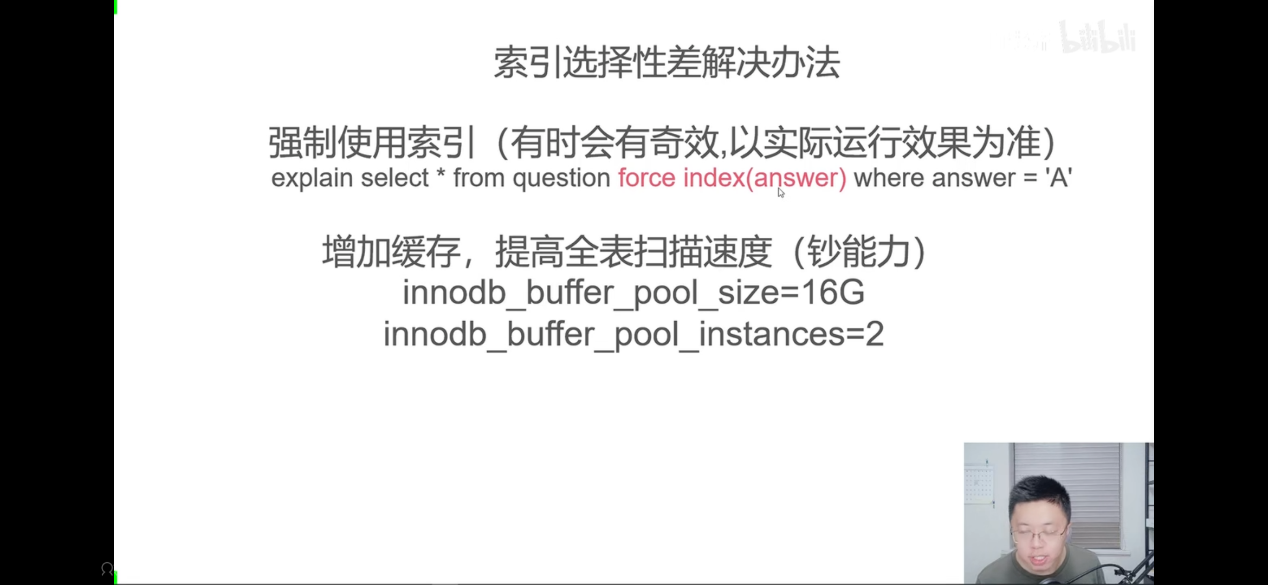
解决方案 组合索引



专业的搜索引擎也能解决这个问题

更换数据源

增加搜索引擎过后 要额外考虑程序复杂性 数据一致性的问题

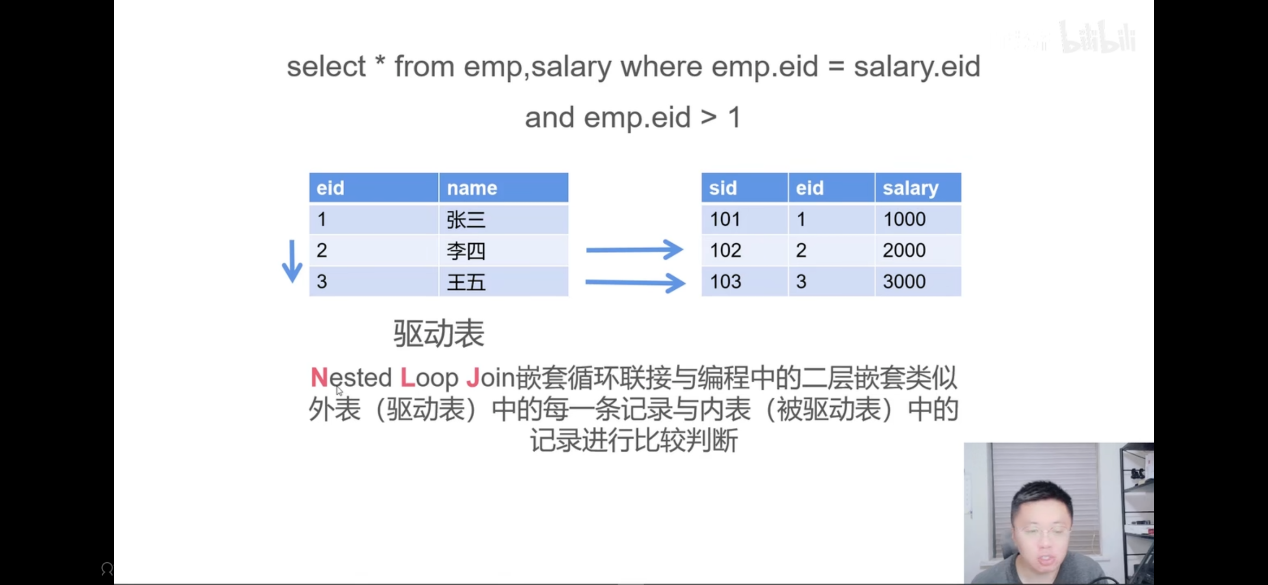


方案二 强制使用索引

mysql选择优化器可能会抽一下

数据覆盖度不高的也走全表扫描了

## 多表关联优化



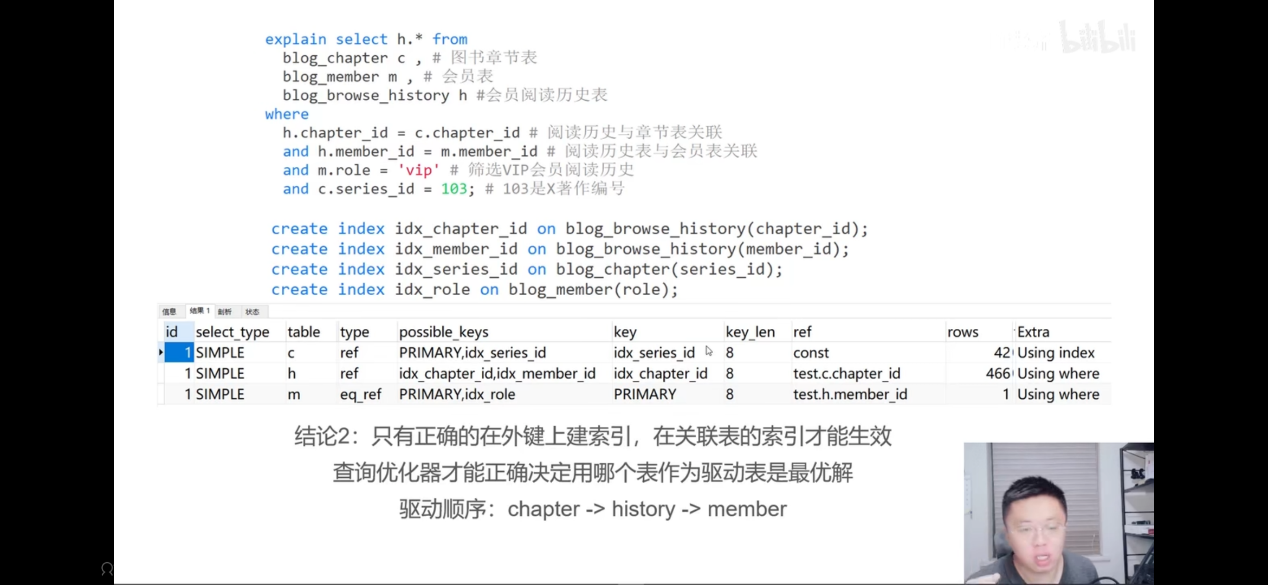
多表关键执行时间



驱动表大小直接决定的关键时处理效率

多表关联 无论是驱动表表还是从表 都要加上索性才有作用

只加上驱动表的索性是没有用的



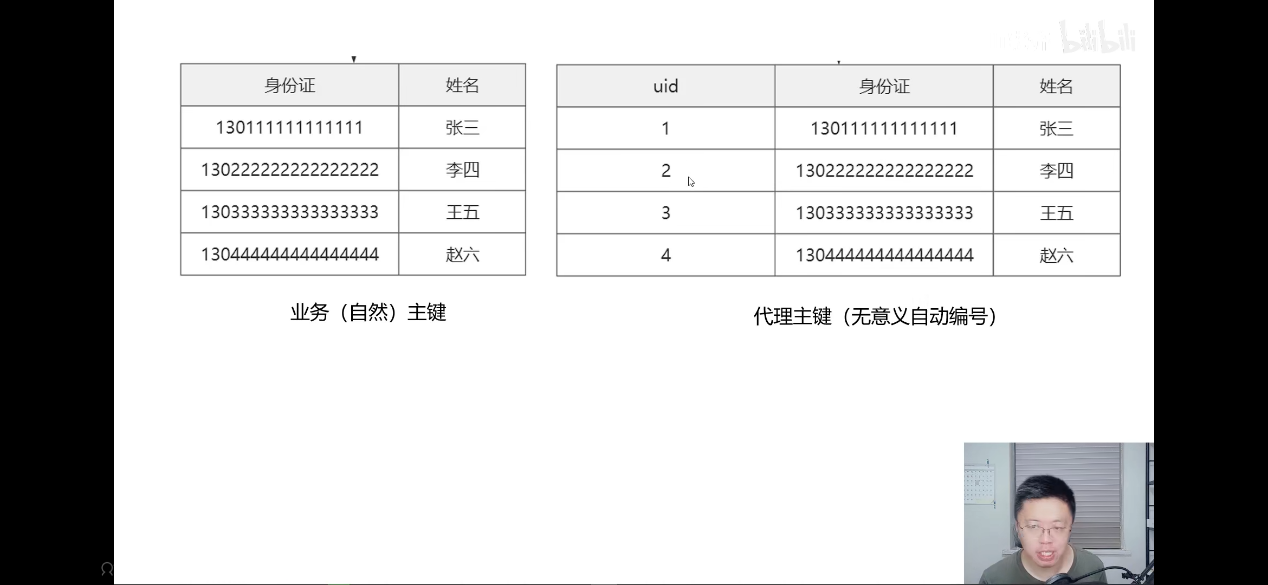
驱动表并不是指定的 而且查询优化器根据索性分布情况选择的

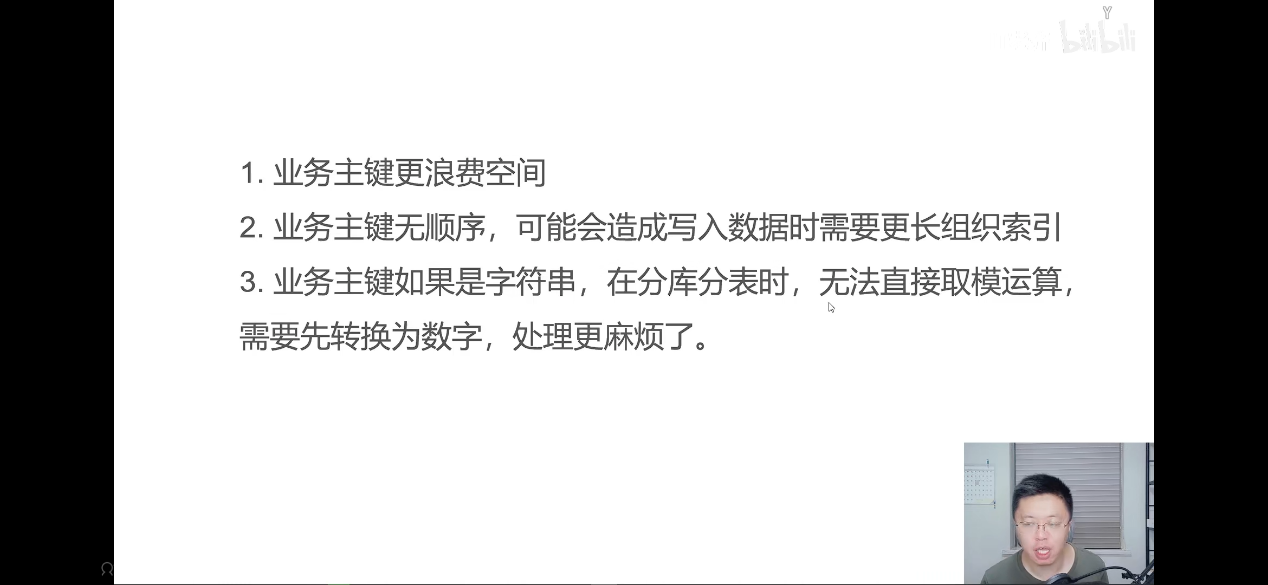
所以很多大厂不建议sql查询关键三张表以上

多表可能导致查询优化器异常 无法正确选择

## 为什么表主键要采用代理主键



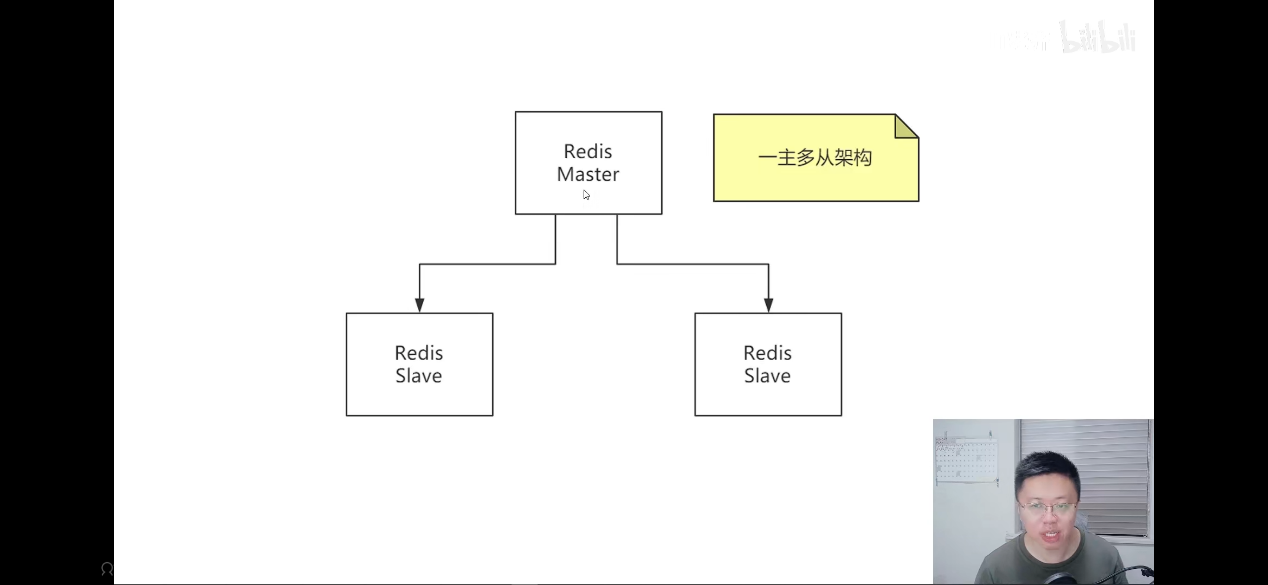




## Redis架构

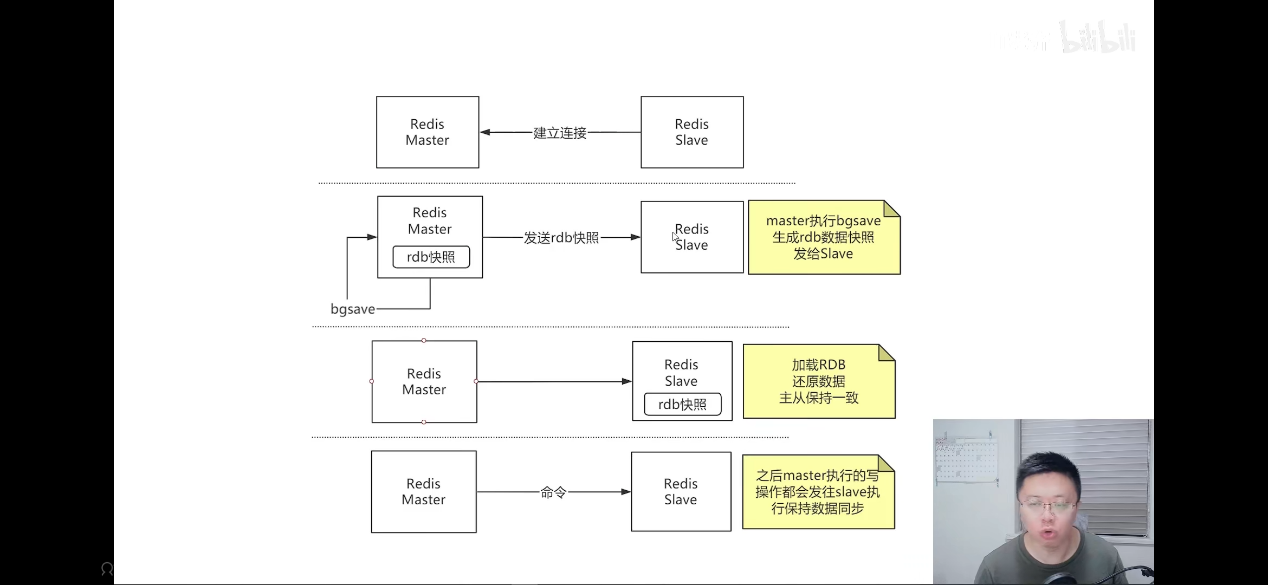


redis高可用

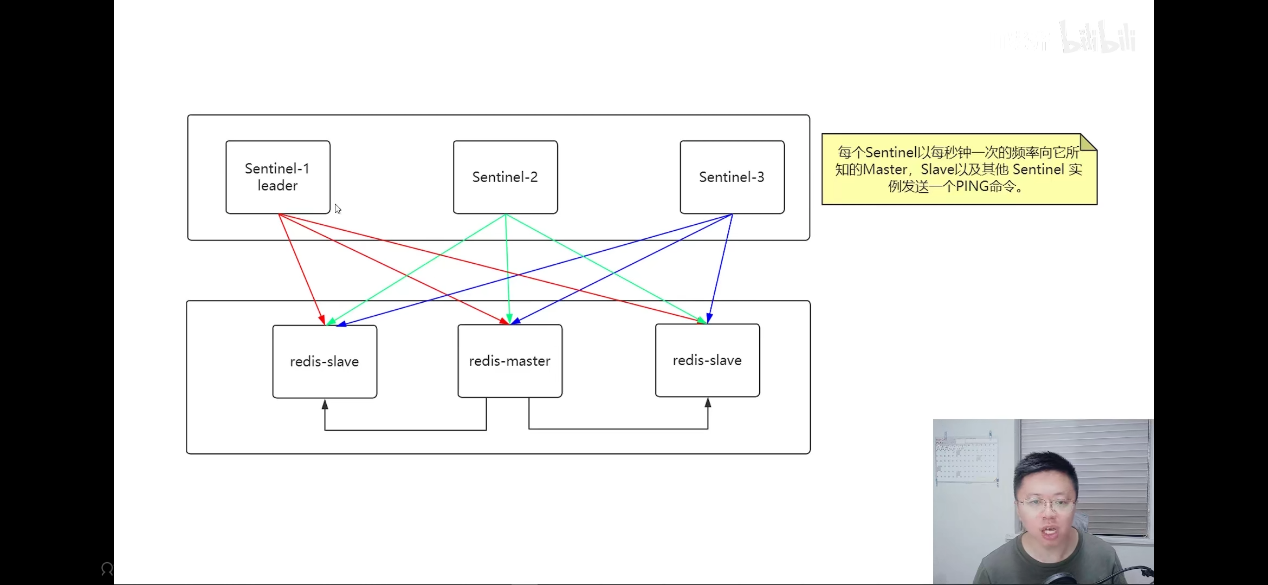


高可用的基础就是构建多个数据相同的节点

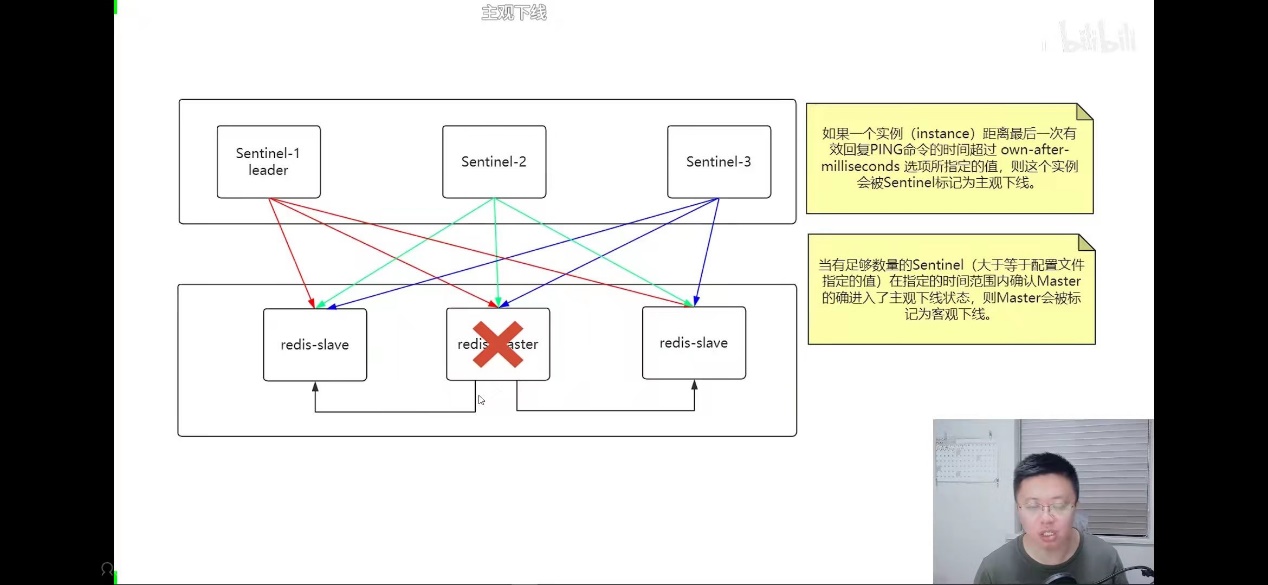
多节点数据传输



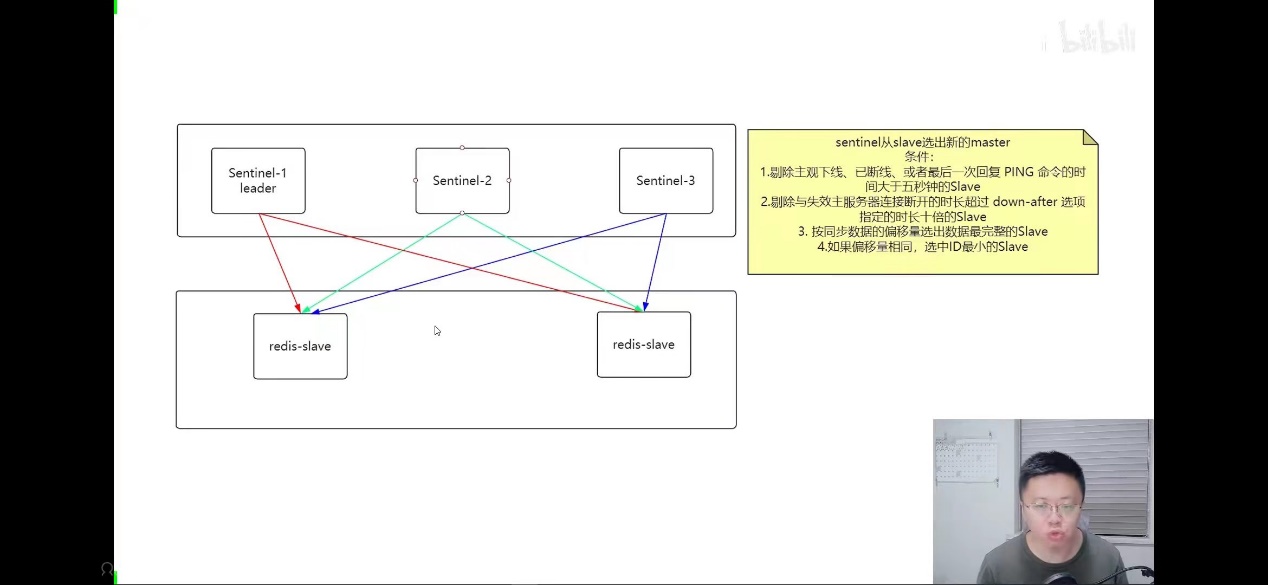
保证高可用最好一个节点一个哨兵



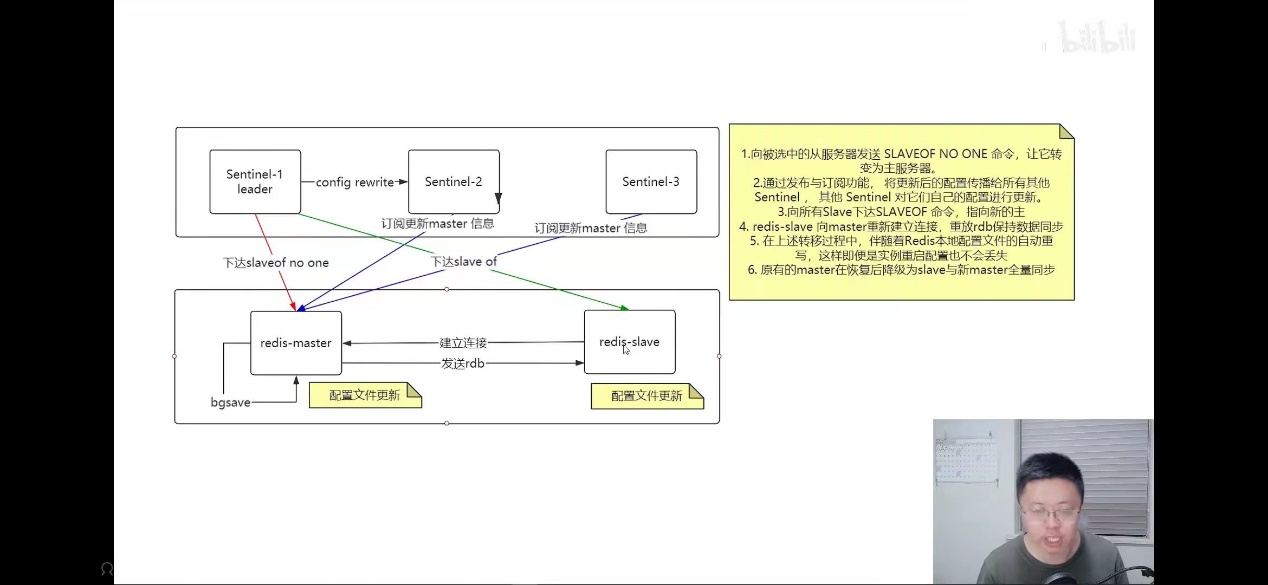
某个sentinel 最后一次ping的时间超过一个特定的值 则会被这个哨兵 标记这个节点为主观下线 当多个哨兵（具体数量由配置文件配）都标记这个下线 那么这个就为客观下线



选新的master 剔除断开时间过长的 选数据偏移量最小的 偏移量相同选id最小的



选出更为更加相近的为新的master



选定新的主节点过后将会由leader发起



raft算法在集群中有超过半数的节点投票 就选为新的节点