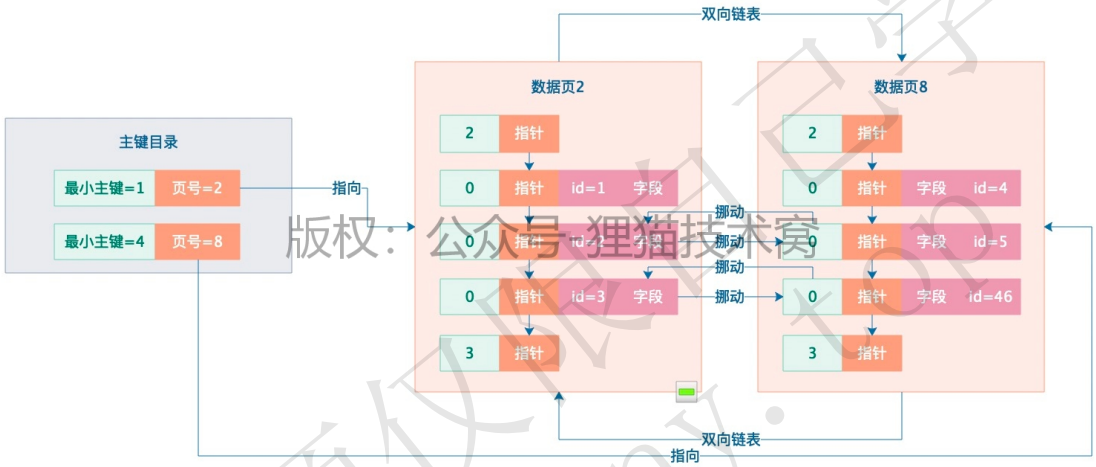
68 索引的页存储物理结构，是如何用B+树来实现的?

索引的页存储物理结构，是如何用B+树来实现的?

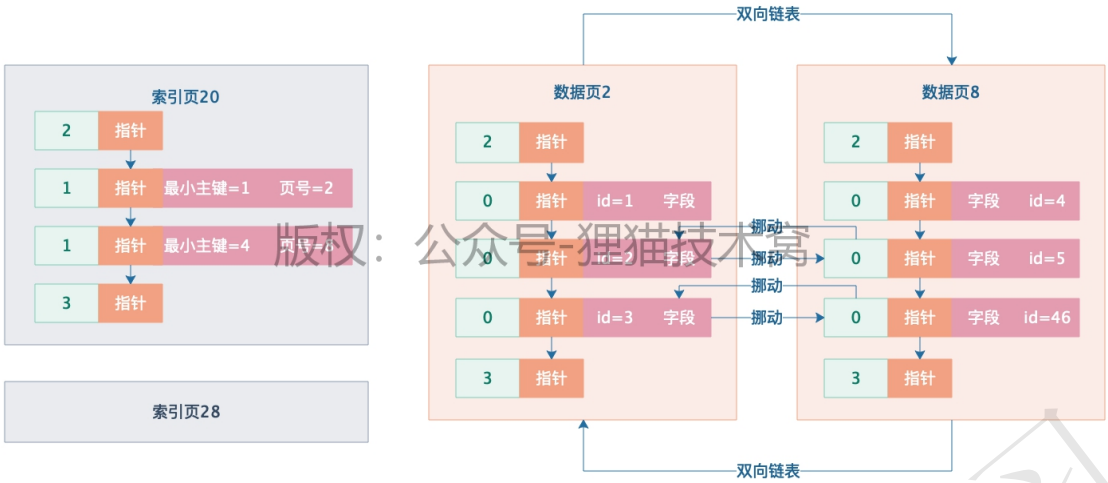
上一次我们给大家说了主键索引的目录结构，只要在一个主键索引里包含每个数据页跟他最小主键值，就可以组成一个索引目录，然后后续你查询主键值，就可以在目录里二分查找直接定位到那条数据所属的数据页，接着到数据页里二分查找定位那条数据就可以了，如下图所示。



但是现在问题来了，你的表里的数据可能很多很多，比如有几百万，几千万，甚至单表几亿条数据都是有可能的，所以此时你可能有大量的数据页，然后你的主键目录里就要存储大量的数据页和最小主键值，这怎么行呢?

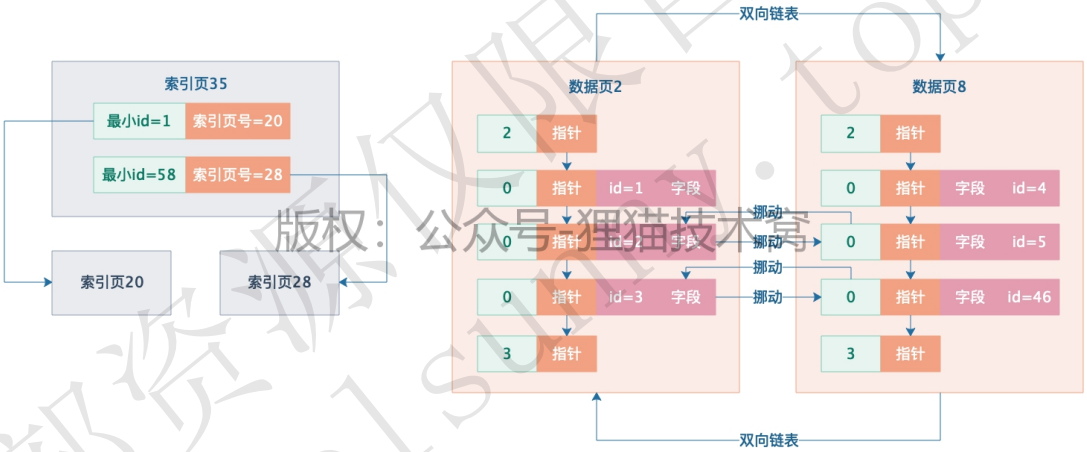
所以在考虑这个问题的时候，实际上是采取了一种把索引数据存储在数据页里的方式来做的

也就是说，你的表的实际数据是存放在数据页里的，然后你表的索引其实也是存放在页里的，此时索引放在页里之后，就会有索引页，假设你有很多很多的数据页，那么此时你就可以有很多的索引页，此时如下图所示。



但是现在又会存在一个问题了，你现在有很多索引页，但是此时你需要知道，你应该到哪个索引页里去找你的主键数据，是索引页20？还是索引页28？这也是个大问题

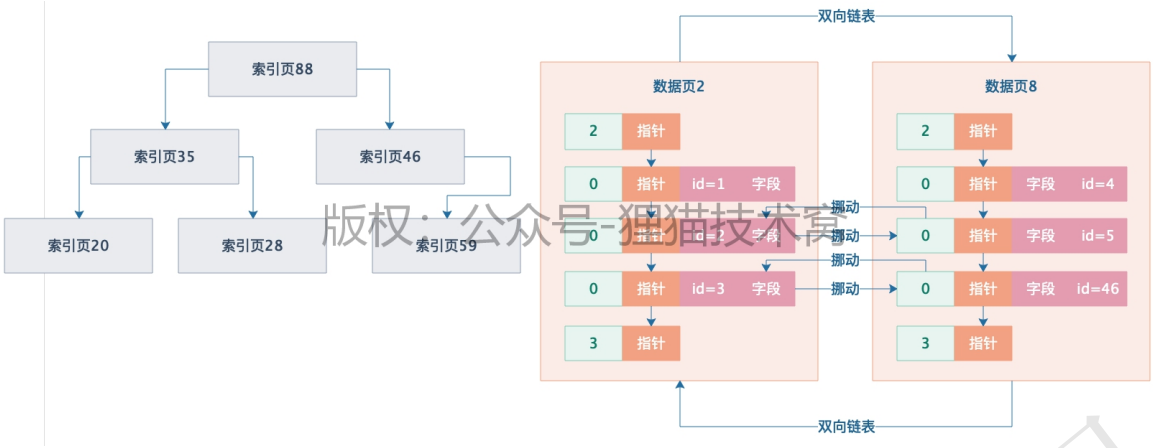
于是接下来我们又可以把索引页多加一个层级出来，在更高的索引层级里，保存了每个索引页和索引页里的最小主键值，如下图所示。



现在就好了，假设我们要查找id=46的，直接先到最顶层的索引页35里去找，直接通过二分查找可以定位到下一步应该到索引页20里去找，接下来到索引页20里通过二分查找定位，也很快可以定位到数据应该在数据页8里，再进入数据页8里，就可以找到id=46的那行数据了。

那么现在问题再次来了，假如你最顶层的那个索引页里存放的下层索引页的页号也太多了，怎么办呢？

此时可以再次分裂，再加一层索引页，比如下面图里那样子，大家看看下图。



不知道大家有没有发现索引页不知不觉中组成了多个层级，搞的是不是有点像一棵树？

没错了，**这就是一颗B+树**，属于数据结构里的一种树形数据结构，所以一直说MySQL的索引是用B+树来组成的，其实就是这个意思。

我们就以最简单最基础的主键索引来举例，当你为一个表的主键建立起来索引之后，其实这个主键的索引就是一颗B+树，然后当你要根据主键来查数据的时候，直接就是从B+树的顶层开始二分查找，一层一层往下定位，最终一直定位到一个数据页里，在数据页内部的目录里二分查找，找到那条数据。

这就是索引最真实的物理存储结构，采用跟数据页一样的页结构来存储，一个索引就是很多页组成的一颗B+树。

好了，今天讲完之后，基本上就初步让大家对索引这个东西有一个入门了，接下来我们就要比较深入的去分析各种索引的物理存储的原理

理解了索引，后续再讲查询原理和执行计划，你基本就很容易理解了。因为其实查询的过程，就是利用各种不同的索引去搜索数据的过程。

End