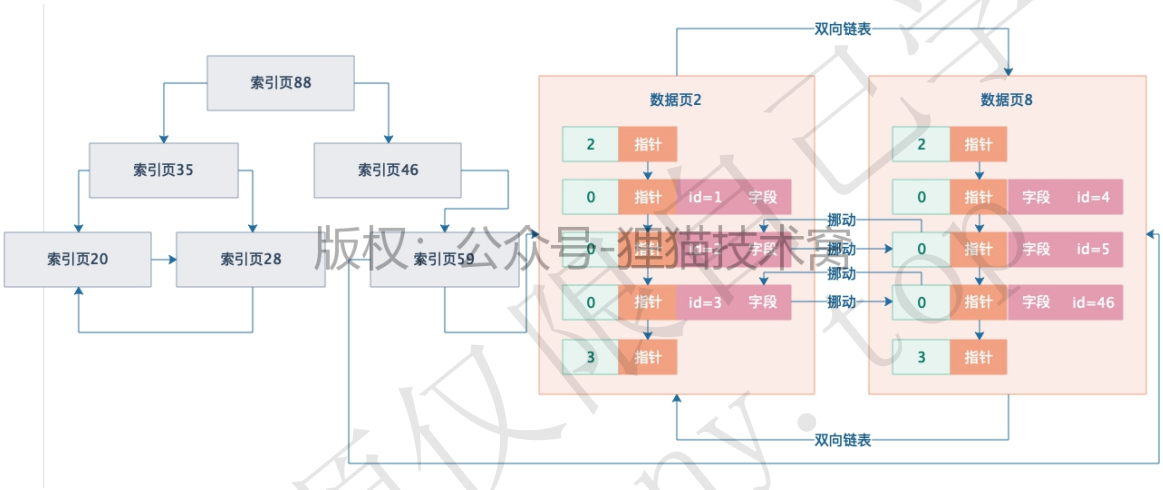
70 针对主键之外的字段建立的二级索引，又是如何运作的?

**针对主键之外的字段建立的二级索引，又是如何运作的?**

---

上一次我们已经给大家彻底讲透了聚簇索引这个东西，其实聚簇索引就是innodb存储引擎默认给我们创建的一套基于主键的索引结构，而且我们表里的数据就是直接放在聚簇索引里的，作为叶子节点的数据页，如下图。





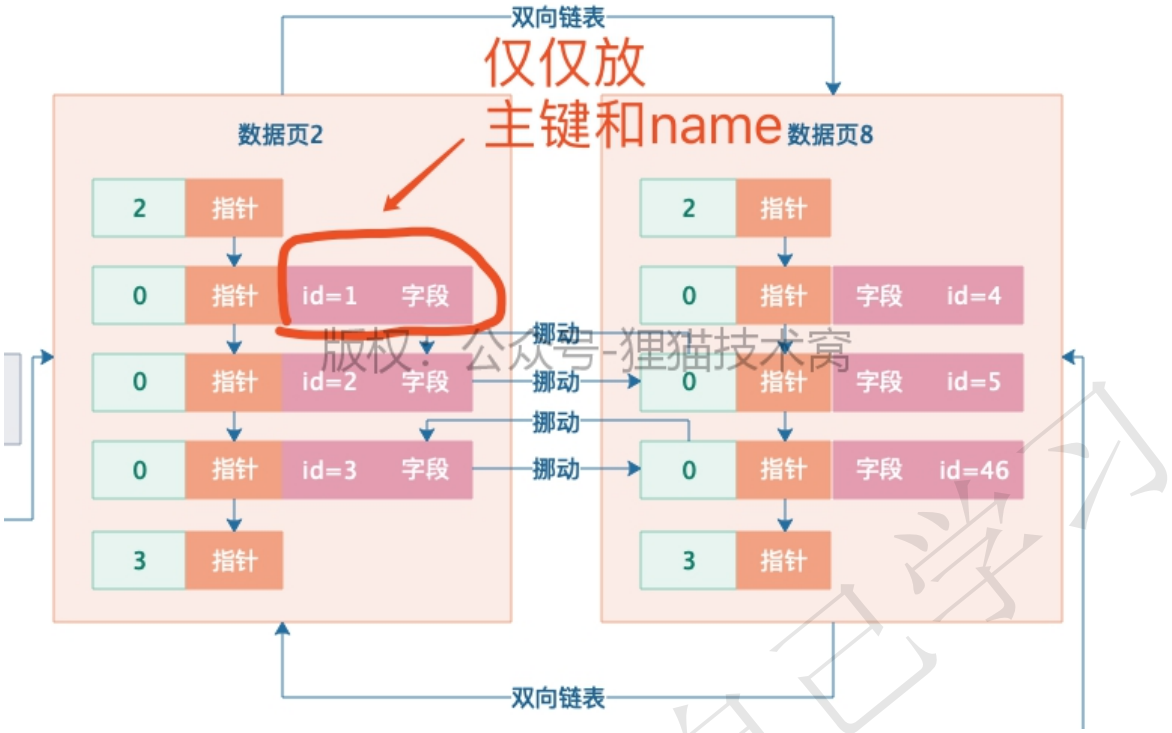

而且我们现在也对基于主键的数据搜索非常清晰了，其实就是从聚簇索引的根节点开始进行二分查找，一路找到对应的数据页里，基于页目录就直接定位到主键对应的数据就可以了，这个其实很好理解。

但是接着我们又会提另外一个疑惑了，那就是如果我们想要对其他的字段建立索引，甚至是基于多个字段建立联合索引，此时这个索引结构又是如何的呢?

今天就给大家讲讲**对主键外的其他字段建立索引的原理。**

其实假设你要是针对其他字段建立索引，比如name、age之类的字段，这都是一样的原理，简单来说，比如你插入数据的时候，一方面会把完整数据插入到聚簇索引的叶子节点的数据页里去，同时维护好聚簇索引，另一方面会为你其他字段建立的索引，重新再建立一颗B+树。

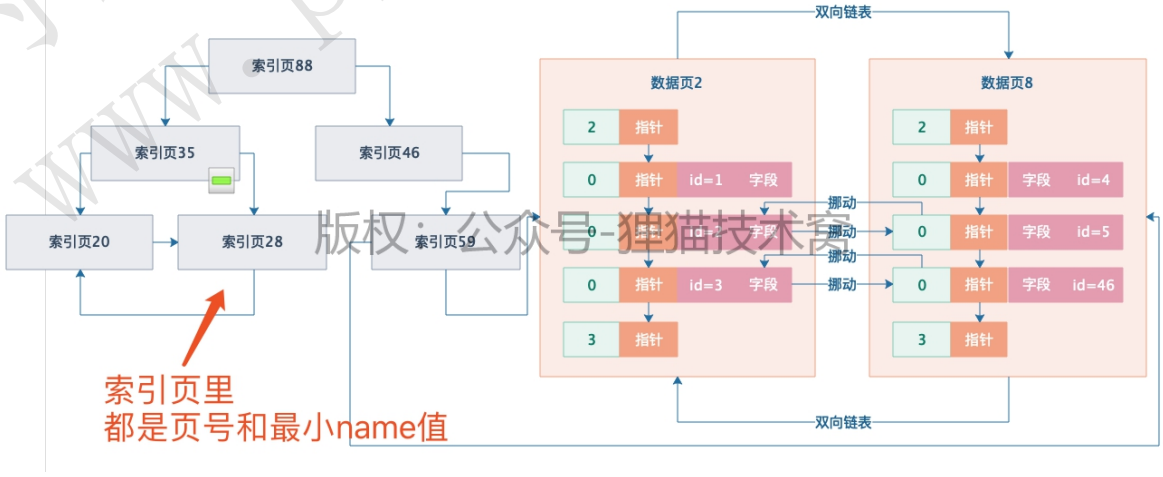
比如你基于name字段建立了一个索引，那么此时你插入数据的时候，就会重新搞一颗B+树，B+树的叶子节点也是数据页，但是这个数据页里仅仅放主键字段和name字段，大家看下面的示意图。




大家注意，这可是独立于聚簇索引之外的另外一个索引B+树了，严格来说是name字段的索引B+树，所以在name字段的索引B+树里，叶子节点的数据页里仅仅放主键和name字段的值，至于排序规则之类的，都是跟以前说的一样的。

也就是说，name字段的索引B+树里，叶子节点的数据页中的name值都是按大小排序的，同时下一个数据页里的name字段值都大于上一个数据页里的name字段值，这个整体的排序规则都跟聚簇索引按照主键的排序规则是一样的。

然后呢，name字段的索引B+树也会构建多层级的索引页，这个索引页里存放的就是下一层的页号和最小name字段值，整体规则都是一样的，只不过存放的都是name字段的值，根据name字段值排序罢了，看下图。




所以假设你要根据name字段来搜索数据，那搜索过程简直都一样了，不就是从name字段的索引B+树里的根节点开始找，一层一层往下找，一直找到叶子节点的数据页里，定位到name字段值对应的主键值。

然后呢？此时针对select * from table where name='xx'这样的语句，你先根据name字段值在name字段的索引B+树里找，找到叶子节点也仅仅可以找到对应的主键值，而找不到这行数据完整的所有字段。

所以此时还需要进行“**回表**”，这个回表，就是说还需要根据主键值，再到聚簇索引里从根节点开始，一路找到叶子节点的数据页，定位到主键对应的完整数据行，此时才能把select *要的全部字段值都拿出来。

因为我们根据name字段的索引B+树找到主键之后，还要根据主键去聚簇索引里找，所以一般把name字段这种普通字段的索引称之为二级索引，一级索引就是聚簇索引，这就是普通字段的索引的运行原理。

其实我们也可以把多个字段联合起来，建立联合索引，比如name+age

此时联合索引的运行原理也是一样的，只不过是建立一颗独立的B+树，叶子节点的数据页里放了id+name+age，然后默认按照name排序，name一样就按照age排序，不同数据页之间的name+age值的排序也如此。

然后这个name+age的联合索引的B+树的索引页里，放的就是下层节点的页号和最小的name+age的值，以此类推，所以当你根据name+age搜索的时候，就会走name+age联合索引的这颗B+树了，搜索到主键，再根据主键到聚簇索引里去搜索。

以上，就是innodb存储引擎的索引的完整实现原理了，其实大家一步一步看下来，会发现索引这块知识也没那么难，不过就是建立B+树，根据B+树一层一层二分查找罢了，然后不同的索引就是建立不同的B+树，然后你增删改的时候，一方面在数据页里更新数据，一方面就是维护你所有的索引。

后续查询，你就要尽量根据索引来查询。

**End**