66 不断在表中插入数据时，物理存储是如何进行页分裂的?

不断在表中插入数据时，物理存储是如何进行页分裂的?

上回我们讲到了数据页的物理存储结构，数据页之间是组成双向链表的，数据页内部的数据行是组成单向链表的，每个数据页内根据主键做了一个页目录

然后一般来说，你没有索引的情况下，所有的数据查询，其实在物理层面都是全表扫描，依次扫描每个数据页内部的每个数据行。

上述描述，其实就是没有索引情况下在一个表中的数据查询情况，这个速度可以说是慢到惊人，所以一般肯定是不能让查询走全表扫描的。因此正常在数据库中的查询，必须要运用到索引来加速查询的执行。

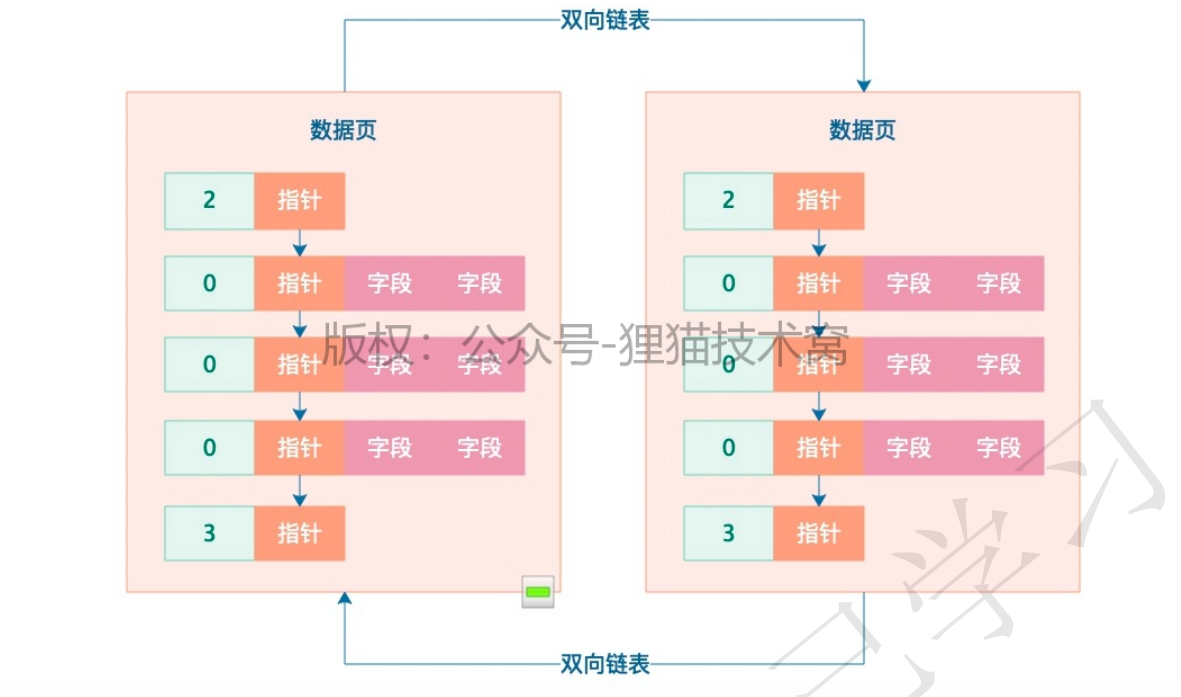
但是今天还是没法直接切入到索引这块内容，因为作为前置知识，今天还得给大家讲解另外一个知识点，就是我们在一个表里不停的插入数据的时候，会涉及到一个页分裂的过程，也就是说，这个表里是如何出现一个又一个的数据页的。

大家都知道，正常情况下我们在一个表里插入一些数据后，他们都会进入到一个数据页里去，在数据页内部，他们会组成一个单向链表，这个数据页内部的单向链表大致如下所示，我们看看

大家看上面的图，里面就是一行一行的数据，刚开始第一行是个起始行，他的行类型是2，就是最小的一行，然后他有一个指针指向了下一行数据，每一行数据都有自己每个字段的值，然后每一行通过一个指针不停的指向下一行数据，普通的数据行的类型都是0，最后一行是一个类型为3的，就是代表最大的一行。

上面就是一个典型的数据页内部的情况，那么今天要讲的页分裂是什么意思呢？

是这样的，假设你不停的在表里插入数据，那么刚开始是不是就是不停的在一个数据页插入数据？接着数据越来越多，越来越多，此时就要再搞一个数据页了，如下图。



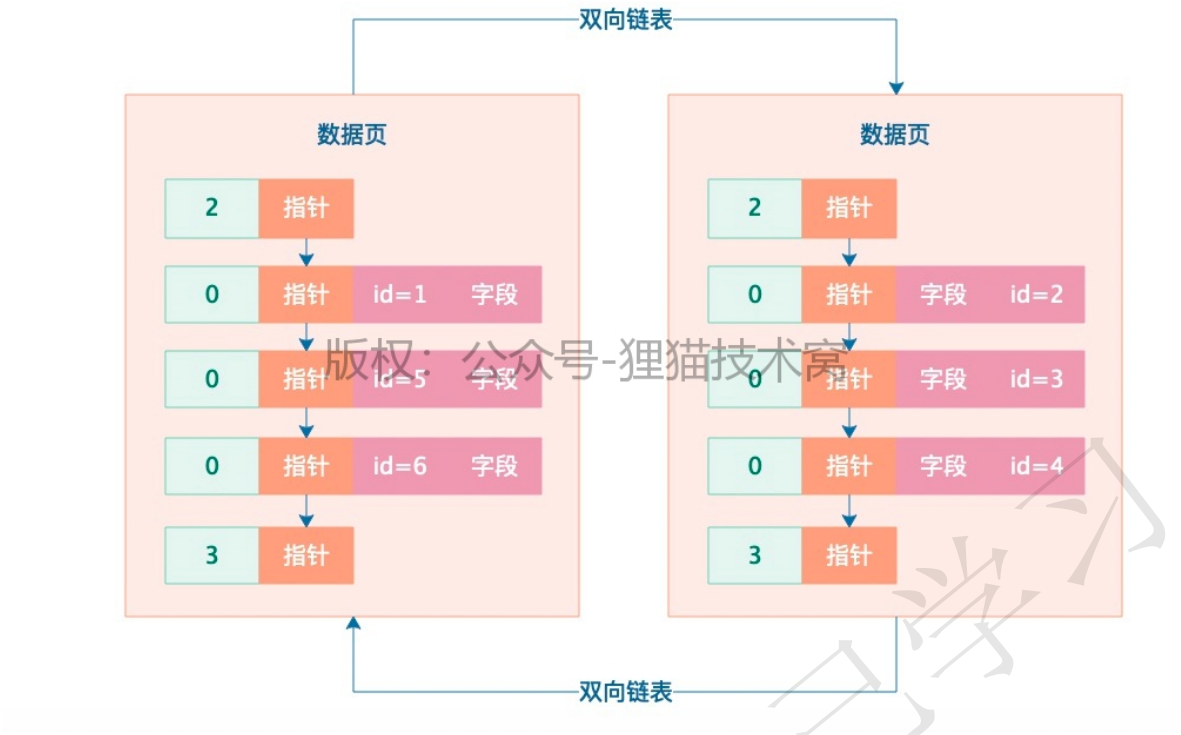
但是此时会遇到一个问题，后续我们会讲到索引这块机制，索引运作的一个核心基础就是要求你后一个数据页的主键值都大于前面一个数据页的主键值，但是如果你的主键是自增的，那还可以保证这一点，因为你新插入后一个数据页的主键值一定都大于前一个数据页的主键值。

但是有时候你的主键并不是自增长的，所以可能会出现你后一个数据页的主键值里，有的主键是小于前一个数据页的主键值的。

比如在第一个数据页里有一条数据的主键是10，第二个数据页里居然有一条数据的主键值是8，那此时肯定有问题了。

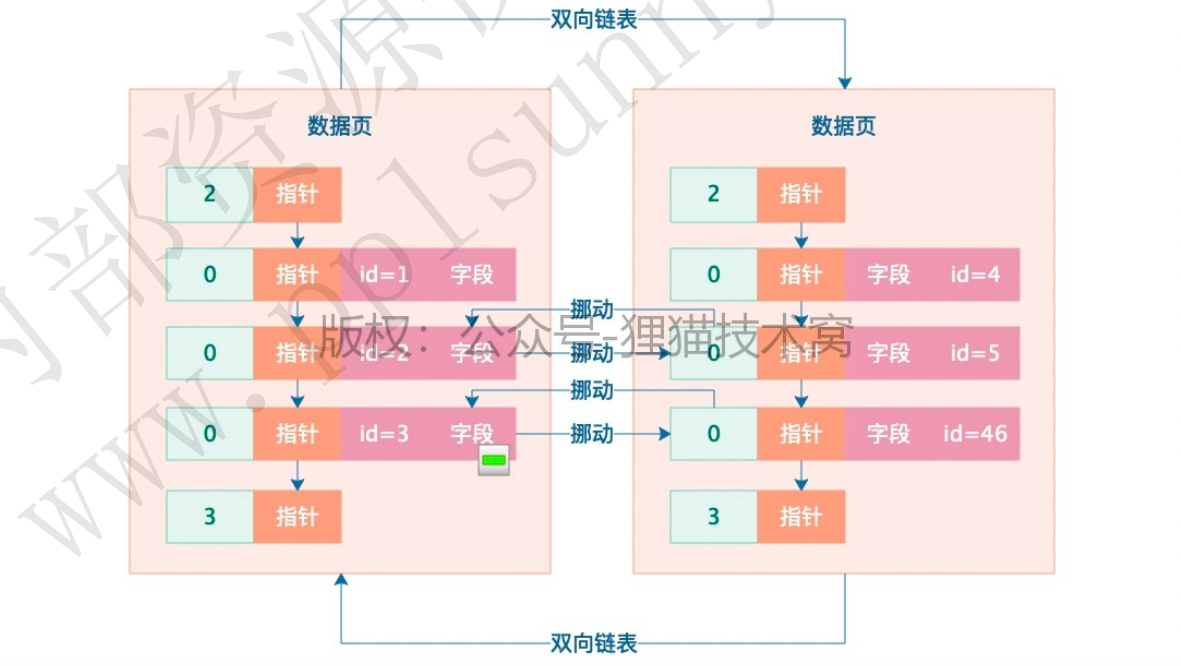
所以此时就会出现一个过程，叫做**页分裂**，就是万一你的主键值都是你自己设置的，那么在增加一个新的数据页的时候，实际上会把前一个数据页里主键值较大的，挪动到新的数据页里来，然后把你新插入的主键值较小的数据挪动到上一个数据页里去，保证新数据页里的主键值一定都比上一个数据页里的主键值大。

大家看下图，假设新数据页里，有两条数据的主键值明显是小于上一个数据页的主键值的，如图所示。



如上图所示，第一个数据页里有1、5、6三条数据，第二个数据页里有2、3、4三条数据，明显第二个数据页里的数据的主键值比第一个数据页里的5和6两个主键都小，所以这个是不行的。

此时就会出现页分裂的行为，把新数据页里的两条数据挪动到上一个数据页，上一个数据页里挪两条数据到新数据页里去，如下图所示。



所以上述就是一个页分裂的过程，核心目标就是保证下一个数据页里的主键值都比上一个数据页里的主键值要大。

这就是今天我们重点要讲的页分裂的过程，有了这个过程，保证了每个数据页的主键值，就能为后续的索引打下基础。

End