什么是索引？

Mysql支持得索引：

B+树索引

全文索引

R树索引

hash索引

创建索引语法：

alter table xx add index indexName(‘xxx’)

create index 两种方式

select \* from employee where emp.no=10000;

索引得作用及效果？？

创建索引

alter table employees add index idx\_birth\_date\_emp(emp\_no,emp\_date);

删除索引

alter table employees drop index idx\_birth\_date\_emp;

索引得数据结构是什么？？

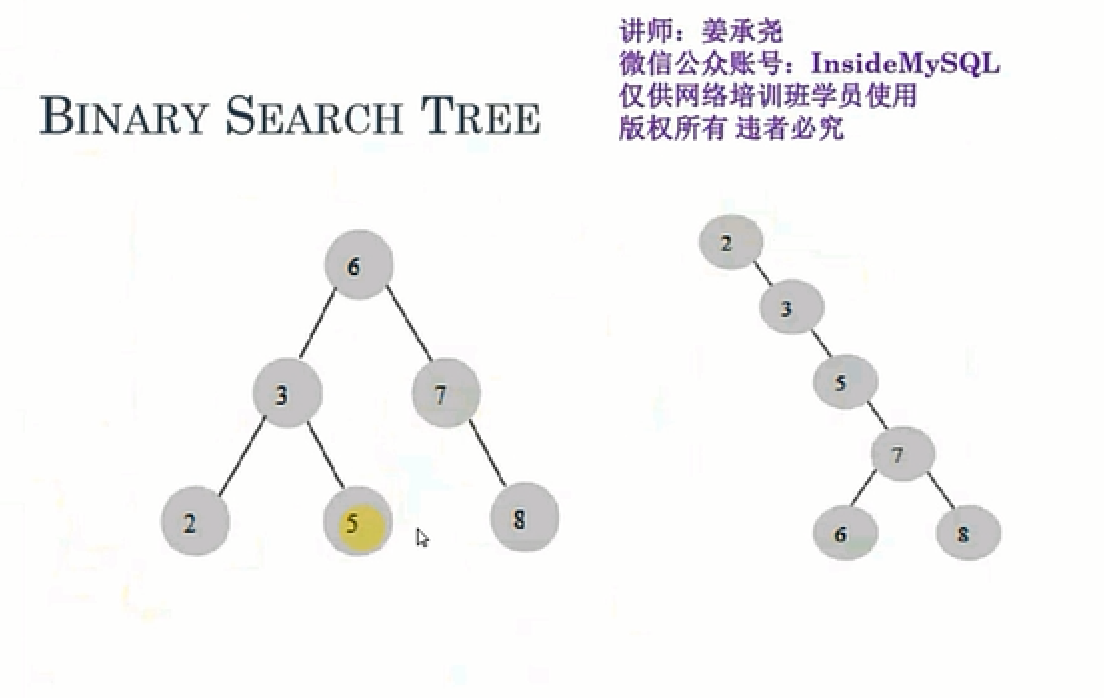
讲解了二叉查找算法。

在已经排序过得数据中进行查询，如果查找得值就是“中间”得这个值，那么查找结束，如果

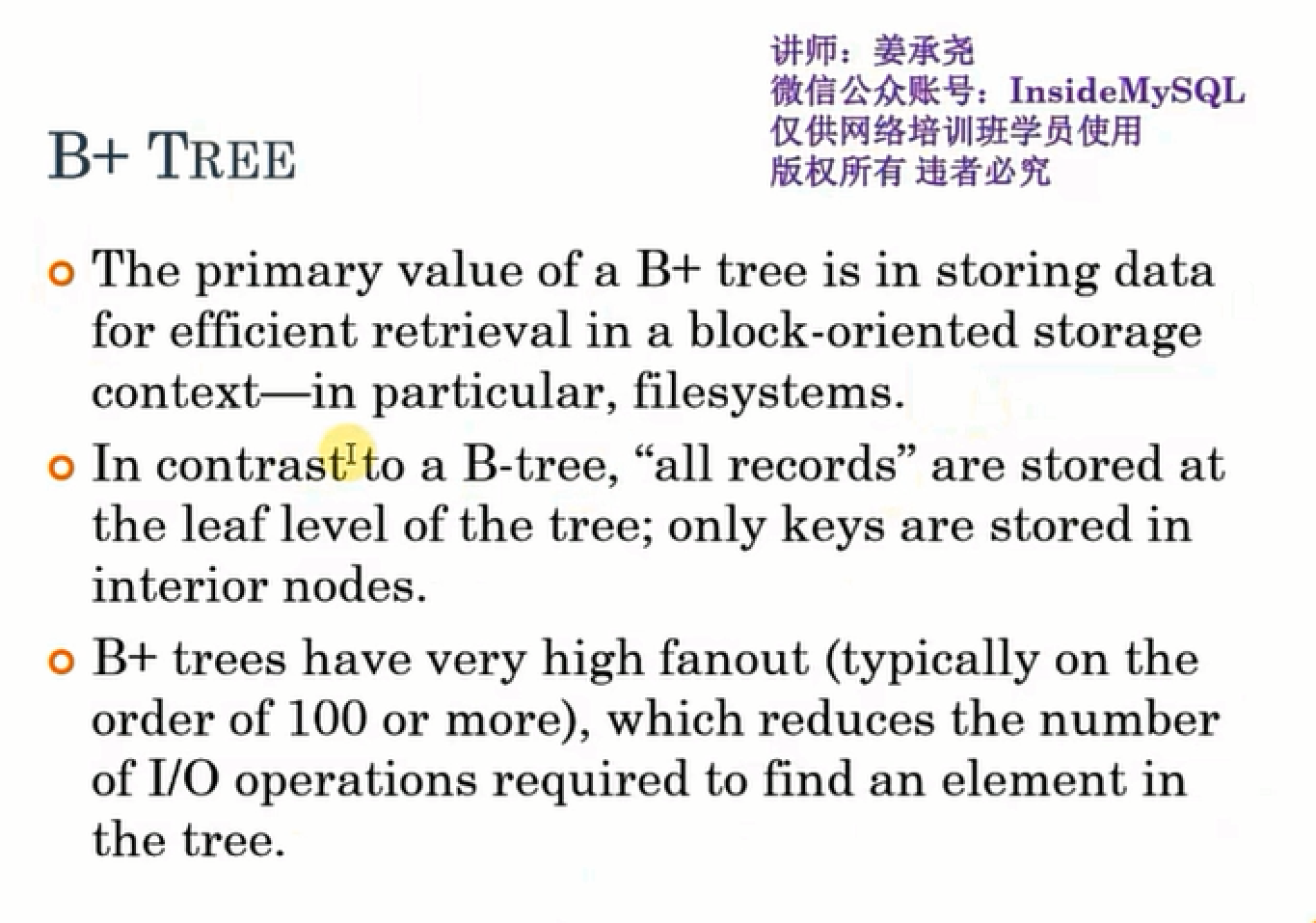
要查找得值小于“中间值”查询左边，如果查找得值大于“中间值”查询右边，直到查询到为止。

二叉树得java实现。

二叉查找法可以过度到二叉树，二叉树的左节点都比中间节点小，右节点都比中间节点大。b+树所有的数据都是存放在叶子节点。非叶子节点存放得是key/point。



B+树的几个名词



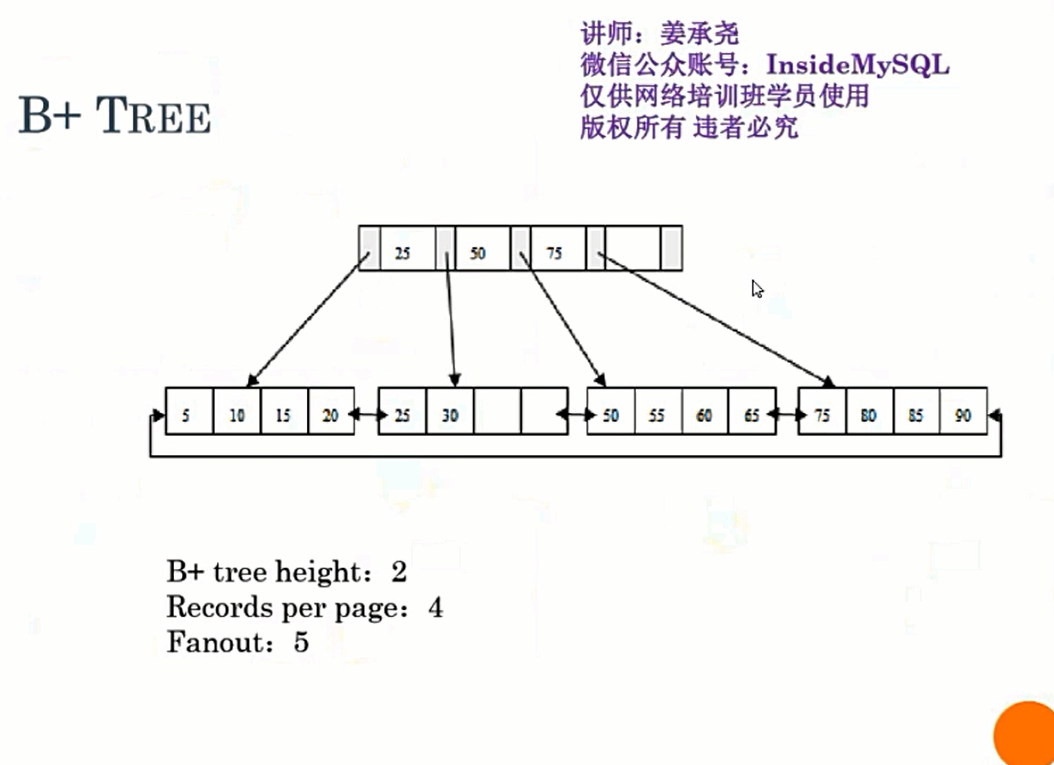
叶子节点

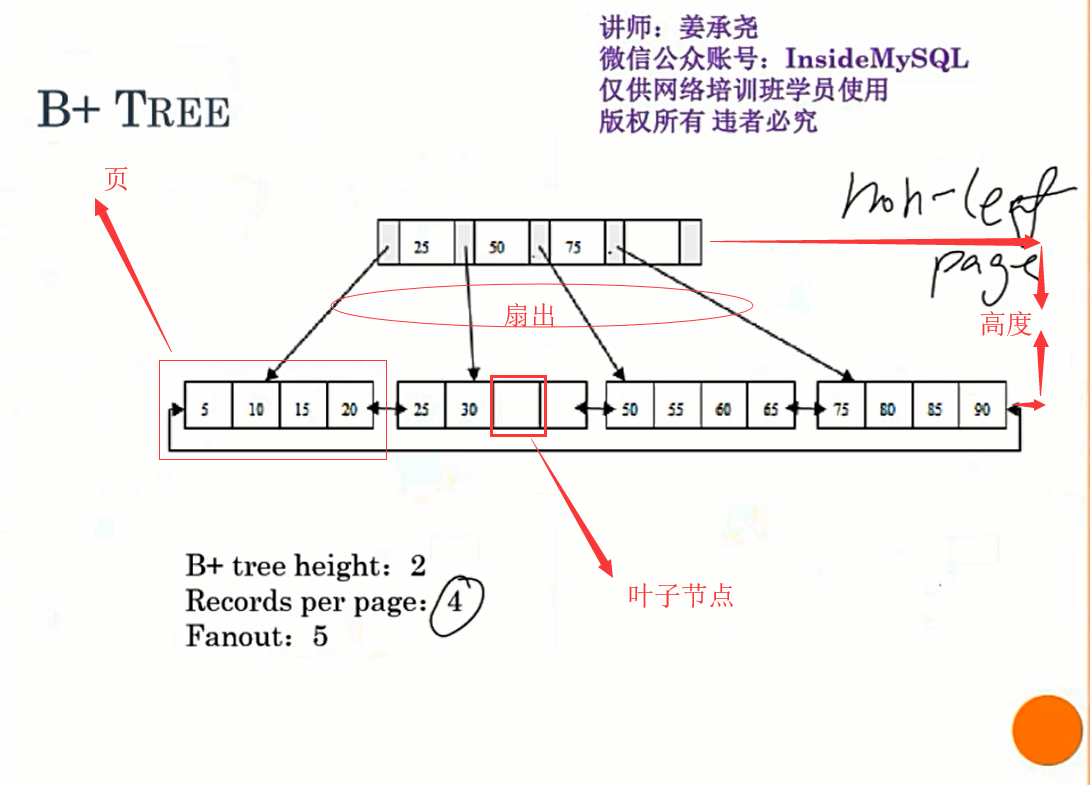
非叶子节点

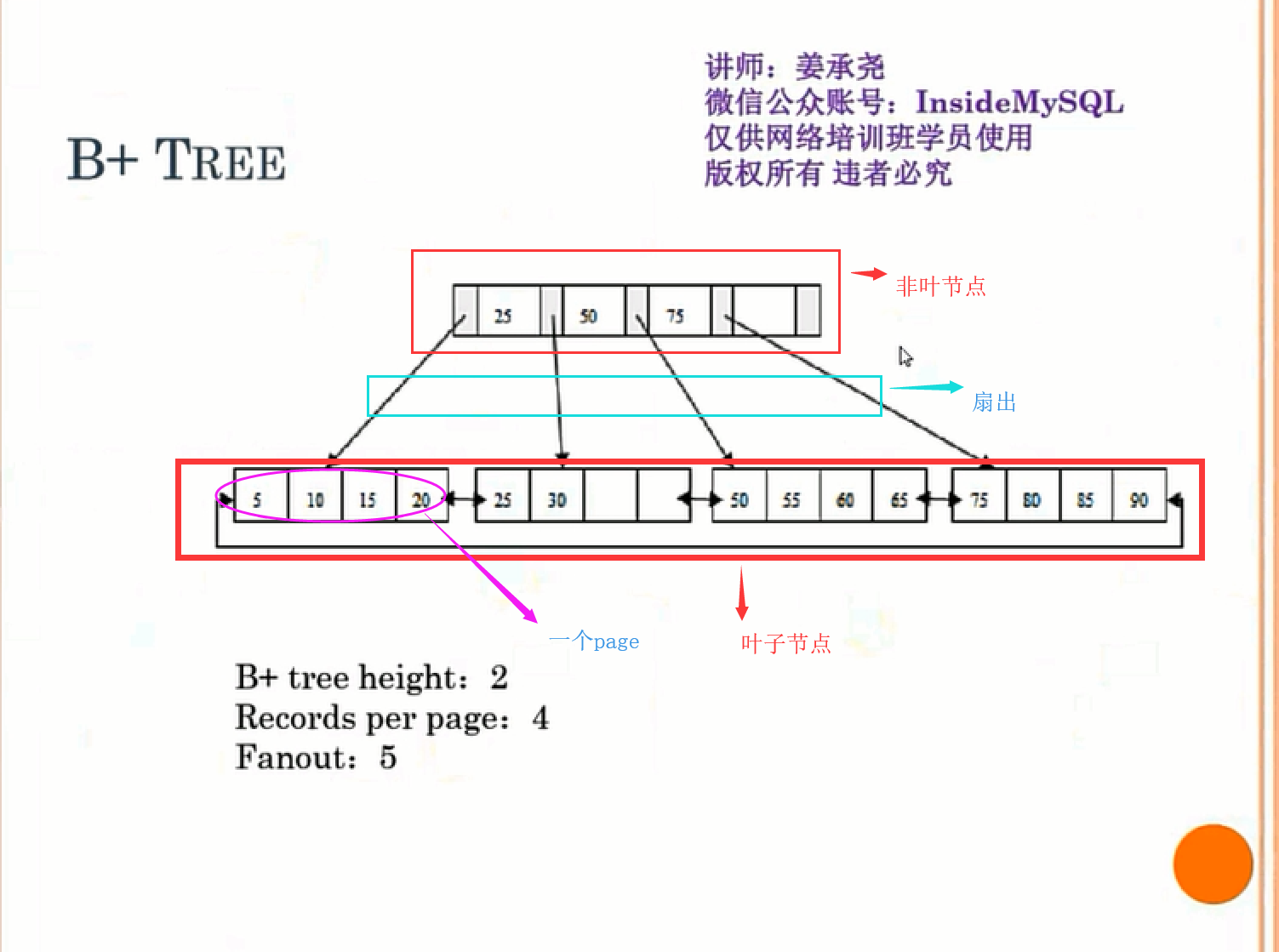
高度

页

扇出：上层节点指向下层节点的指针就是扇出







B+ 树扇出很高，代表什么？ 代表这颗树不会很高

假设：

每一页可以存100条数据，有100个扇出：

当树的高度为1 的时候可存数据： 100（就一层，所以全部都是叶子节点）

当树的高度为2 的时候可存数据： 100（扇出）\*100（每个扇出指向的叶子节点的数据量）

当树的高度为3 的时候可存数据： 100（第一层扇出）\*100（第二层扇出）\*100（每个扇出指向的叶子节点的数据量）

以此类推 。。。。

当树的高度为n=1 的时候客村数据：叶子节点存放得数据量

当树的高度为n>=2 的时候客村数据：（100的n-1次方条数）\*（每一页得数据量）

那么接着说：树的高度小，在查找的数据代表查找的次数就会很少

假设一页有100条数据，有100个扇出， 现在数据量有100w，在查找的时候查找3次就可以找到。

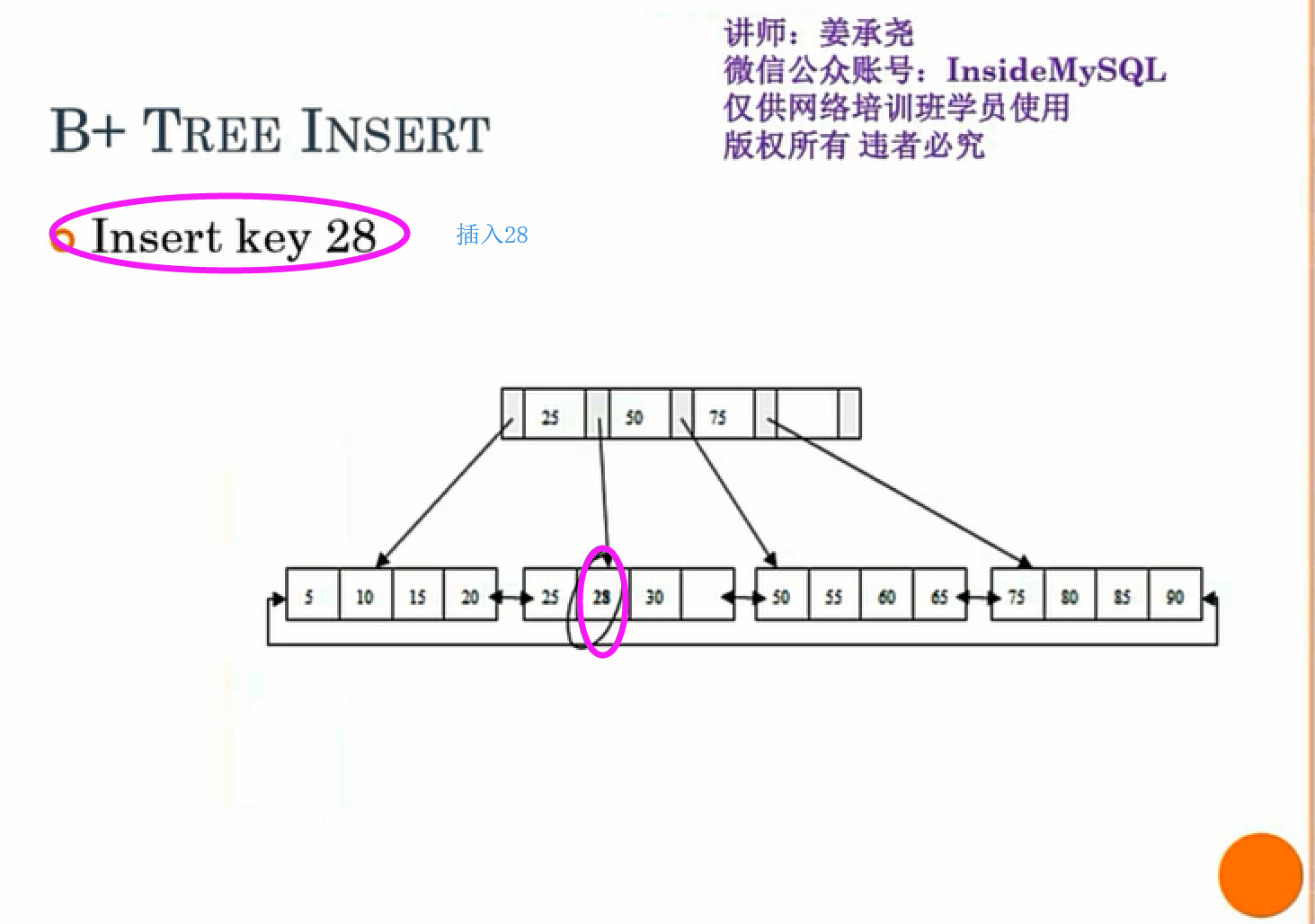
==============================分割线================================

那么平衡二叉树插入数据的时候，二叉树就要进行左旋或者右旋以此来保持平衡，那么在B+树也有这样的操作，

在B+树中叫分裂。分裂算法：

1 选择一个中间节点，以这个中间节点作为分裂点，分裂成两个叶，A叶，B叶。

2 如果分裂出来的A叶，B叶都满了，那么上层节点还要分裂



B+树插入开销很大，但是速度也很快，为什么速度快，因为叶子数据都是排序好的。

B+树查找顺序：

通过B+ 树可以定位到数据所在的页中，再在页内通过二分查找法来查找具体数据。因为在页内的数据都是按照逻辑排序好的，可以很快速的定位数据。

B+树为什么有这么高的扇出呢？

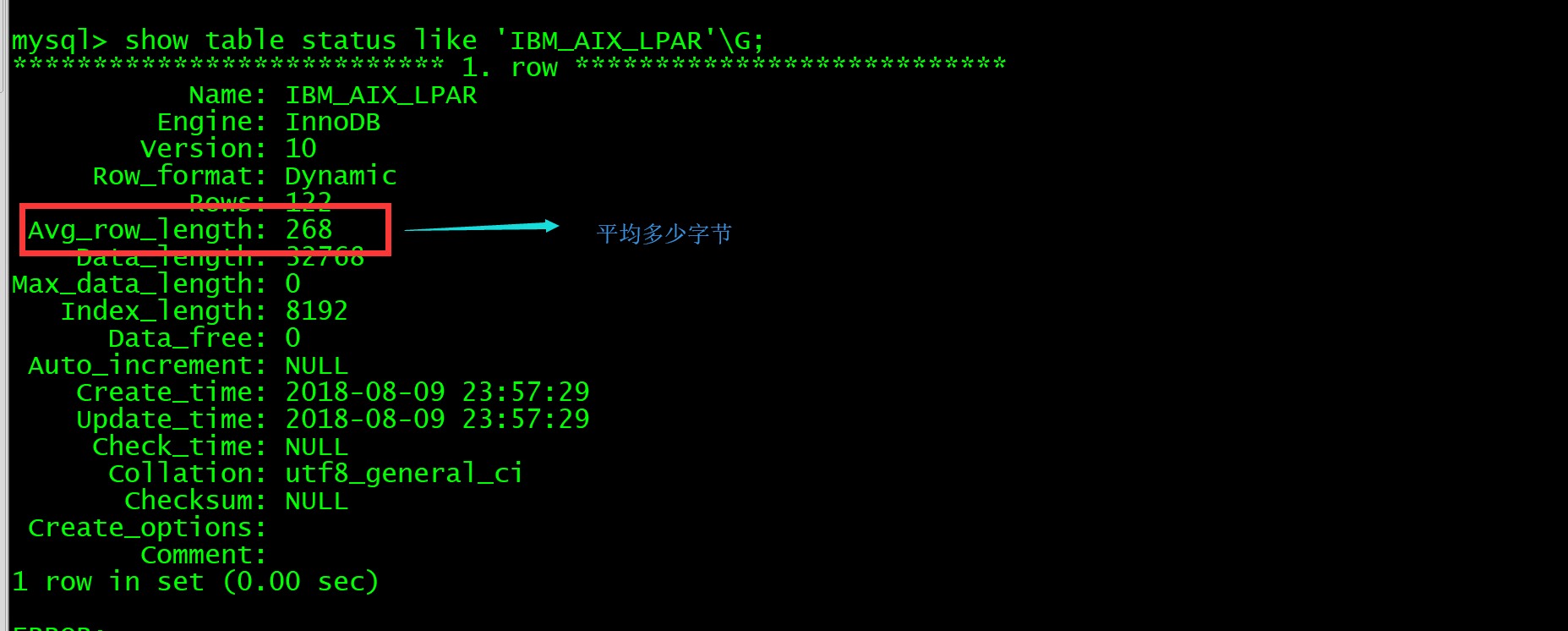
怎么查看一条记录有多大？

在mysql中一个页是16k，树的高度为一，为二，为三 能存多少条数据呢？

1 要看每条数据占用多少存储空间，怎么看一条数据占用多少空间呢？第一可以通过查看创建表的语句来计算，

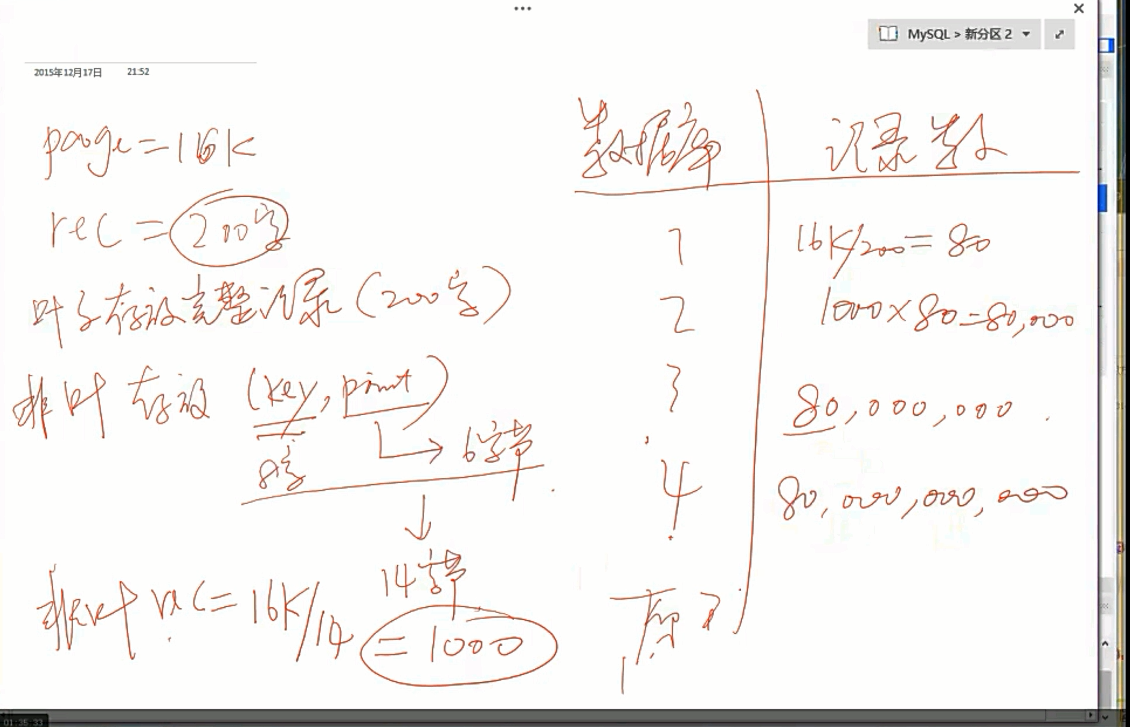
但是遇到了varchar就会计算不准；还可以通过show table status like 'TableName'\G 可以查看，因为有一个

参数是 avg\_row\_length 意思是平均每一条记录得长度。



2 接着上面的问题一个页是16k，树的高度为一，为二，为三 能存多少条数据呢？

数据库叶子节点存放的是一条完整的记录（一条记录假如是200k），非叶子节点存放的是（key/point），其中point是指针，指针固定大小是6个字节，那么这个key是什么呢？讲解语法的时候我们有这样一条语句alter table emp add index indexName(birthday\_date),对那个列进行索引那个列就是key，假设birthday\_date是8个字节，那么非叶节点的一条数据的大小就是8+6 14个字节，**（再mysql innodb那本书最后一页有图，图解释了怎么确定key/point得大小，也就是非叶节点得大小）**。同时也解释了为了扇出高树的高度就矮。树得高度矮在进行查询数据得时候io次数就少，查询时间就快。



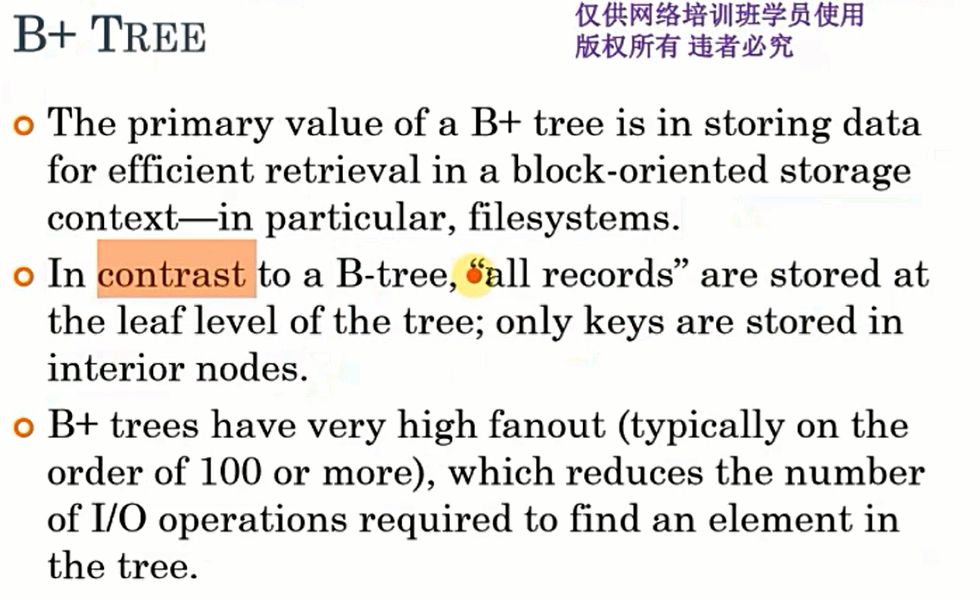
3 树的高度为一，为二，为三又代表什么意思呢？

这里就和io有关系了，当树的高度为三的时候理论上可以存8000w数据，再这些数据里找一条数据，只需要三次io读取三个页就可以查找到，那么查找时间是多少呢？

这里有一个概念是iops，就是每秒多少次io，假设服务器磁盘是100iops，那么读取三次io的时间时间就是3/100秒，

也就是说再树的高度为三，数据量达到理论值（8000w）的时候查找一条记录的时间是3/100秒，如果有cache还会更快。

B树和B+树区别：



填充率：插入，存储时二叉树得填充率不一样。

innodb\_fill\_factor 是插入时得填充率

关于split得问题！！！

当出现树得高度变化得时候，加得锁比较多。一旦树加锁，并发就可能受到影响。所以经可能减少split得情况，

那么怎么减少split呢？

举例：原来数据得大小时20k，但是现在更新了更新到30k，但是现在恰巧数据所在的页存储空间快满了，只有27k了，

那么这条数据就不能原地更新了（原地更新就是在原来位置进行更新，因为原位置得存储空间还够），必须需要其他存储空间来存储这30k得数据，

这个时候就会发生split操作，简单而言：就是页中得数据已经满了，当有数据更新后数据量变大了，数据库就要进行split。所以将页的存储空间不要一开始就占满，占一个百分比即可。

举例:什么情况下数据更新后数据量变大了，比如物流行业一个快递信息的更新。信息量越来越大，就可能出现split得情况。