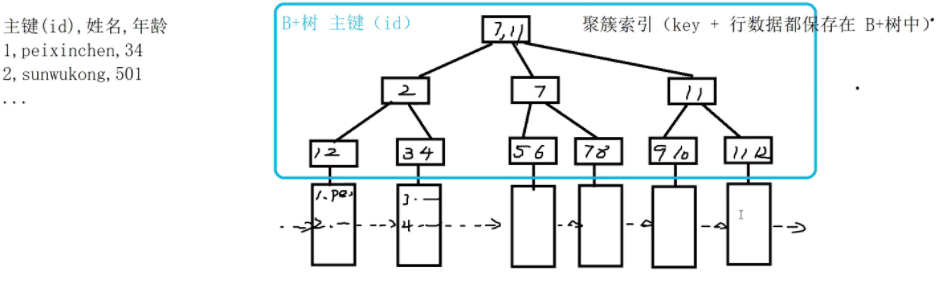
**MySQL内部使用B+树的原因:**

B+树，每层上的key比二叉树要多，所有同样的key的情况下，高度比较低

MySQL的存储一般放在磁盘中，硬盘的IO速度相较于内存比较浪费时间。查找每经过一次高度，就要发生一次读取。为了优化，降低树的高度，选择B+树

**MySQL内部（InnoDB)存储数据：**

将主键作为key,保存成一颗B+树，所有主键+行数据，全部保存在叶子中



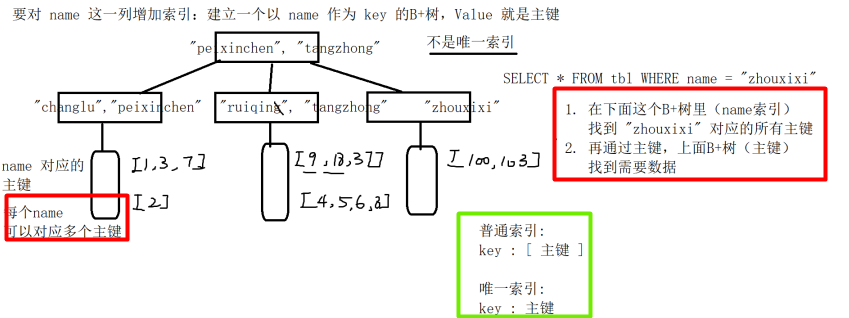
**索引：为了提升效率**

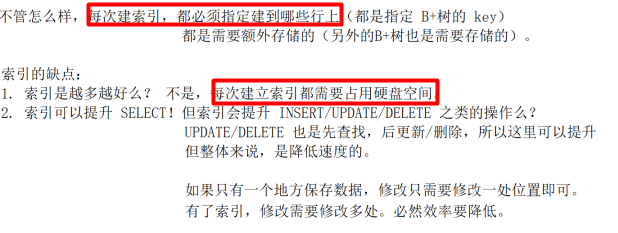
如果没有命中索引，则无法使用B+树，只能进行遍历，时间复杂度是O(n);

提升必须使用搜索专用的数据结构：

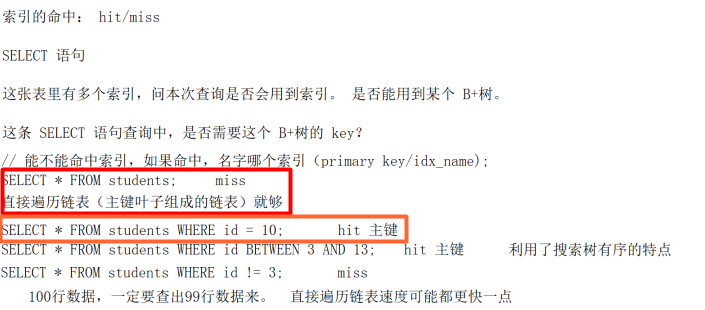
clipboard.png

**下面例子中“上面那个树”指的是上面那个B+树：**



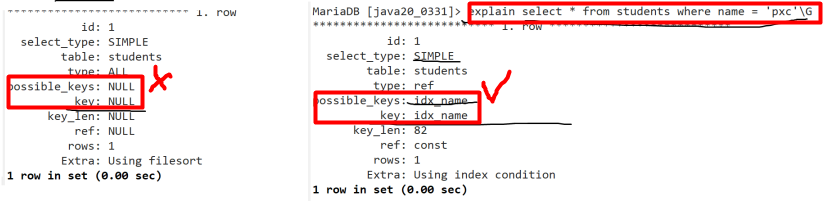


**索引的命中：**



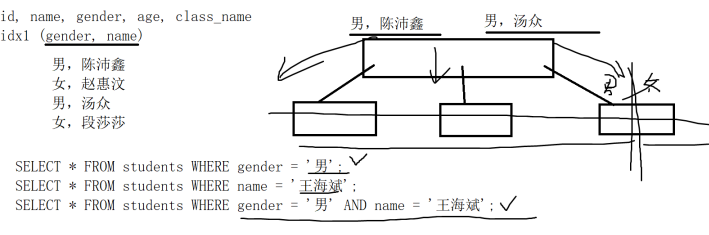
**是否使用索引：只需要判断有无索引的情况下，哪个效率高**

**没必要去猜是否命中索引：使用explain命令进行判断**

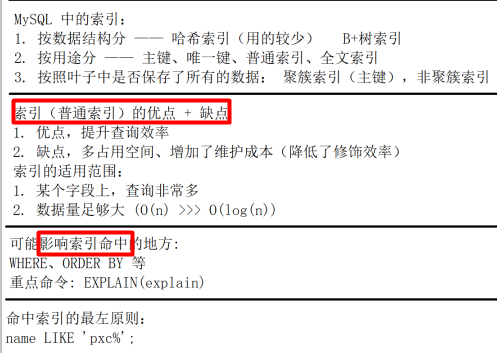


**最左原则（单一/复合索引）：**

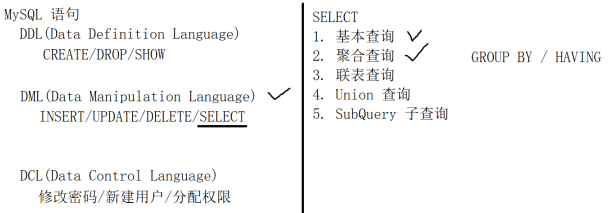
**只有左边可以进行匹配，可以拿着信息在B+树中确定方向**



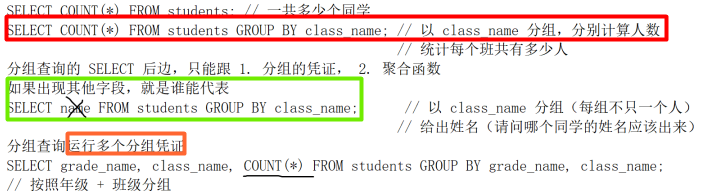
**索引总结：**



**MySQL语句**



**聚合查询——将表中的数据分组看待，每组分别给出结果**

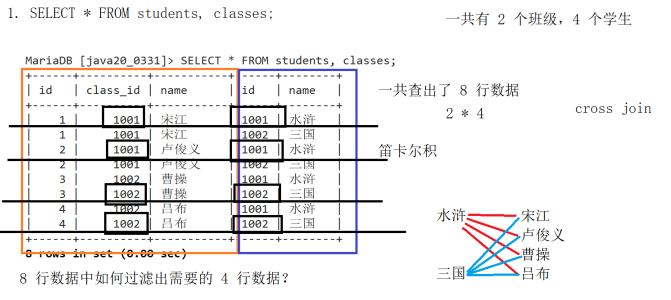


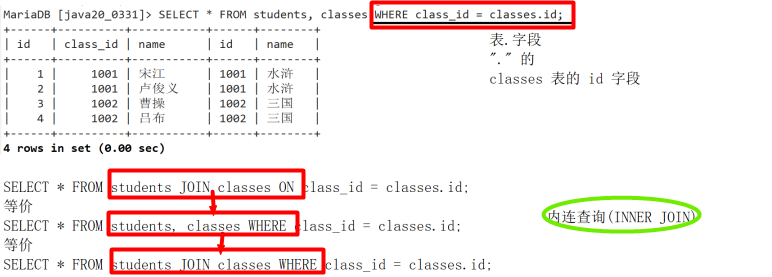
**HAVING在聚合后的结果中进行筛选**

**WHERE在聚合前进行筛选**

WHERE先进行筛选>GROUP BY分组>HAVING对结果进行筛选

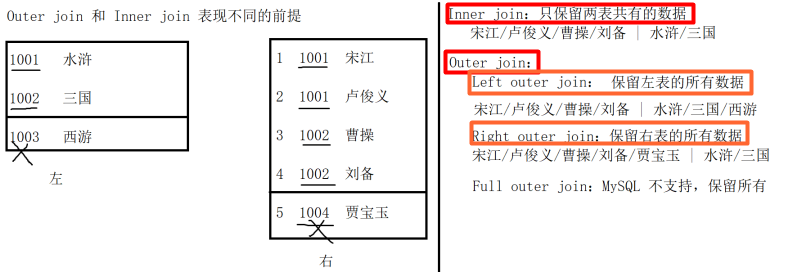
**连表查询：join**



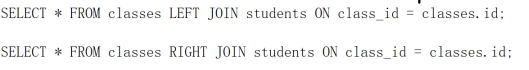


**内连( join)查询与外联(Outer join)查询：**

前提：两张表中有关联字段



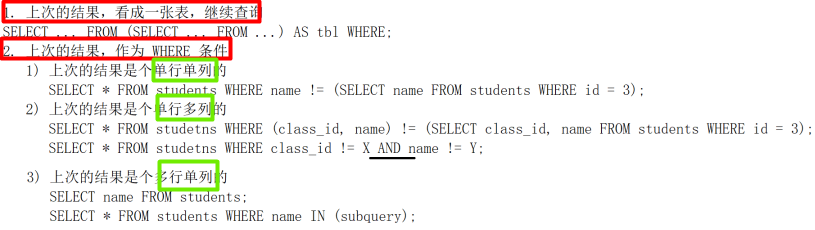
Outer join的左连与右连



**自连查询：相当于多表连表查询，但需要给表起别名，消除歧义**



**子查询：**



**Union查询：**

